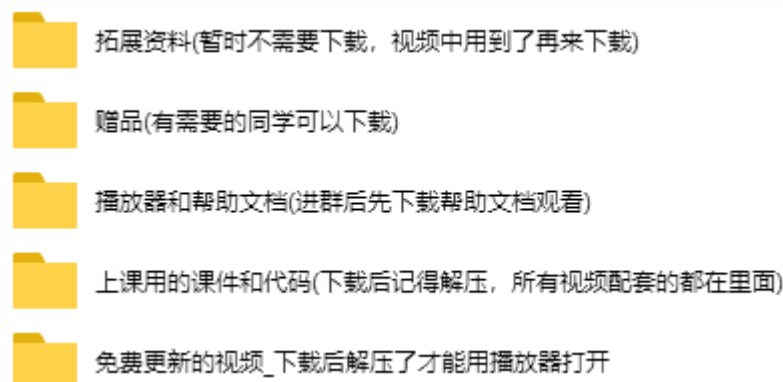


# 第八讲:图论最短路径问题

本讲将简要介绍图论中的基本概念, 并主要讲解图论中的最短路径问题。根据图的不同, 我们将学习两种不同的算法: 迪杰斯特拉Dijkstra算法和Bellman-Ford (贝尔曼-福特) 算法。

## 温馨提示

- (1) 视频中提到的附件可在**售后群的群文件**中下载。  
包括**讲义、代码、我视频中推荐的资料**等。



(2) 关注我的**微信公众号《数学建模学习交流》**，后台发送**“软件”**两个字，可获得常见的建模软件下载方法；发送**“数据”**两个字，可获得建模数据的获取方法；发送**“画图”**两个字，可获得数学建模中常见的画图方法。另外，也可以看看公众号的历史文章，里面发布的都是对大家有帮助的技巧。

(3) **购买更多优质精选的数学建模资料**，可关注我的微信公众号《数学建模学习交流》，在后台发送**“买”**这个字即可进入店铺进行购买。

(4) 视频价格不贵，但价值很高。单人购买观看只需要**58元**，和另外两名队友一起购买人均仅需**46元**，视频本身也是下载到本地观看的，所以请大家**不要侵犯知识产权**，对视频或者资料进行二次销售。

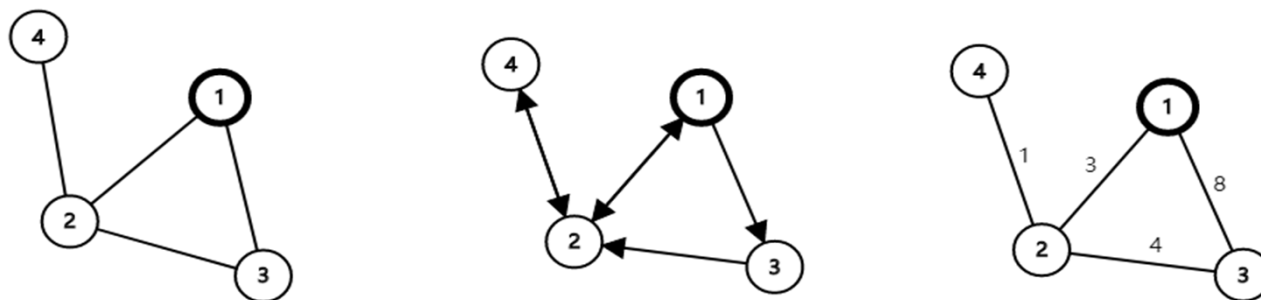
## 图的基本概念

图论中的图 (Graph) 是由若干给定的点及连接两点的线所构成的图形, 这种图形通常用来描述某些事物之间的某种特定关系, 用点代表事物, 用连接两点的线表示相应两个事物间具有这种关系。

一个图可以用数学语言描述为  $G(V(G), E(G))$ 。  $V(\text{vertex})$  指的是图的顶点集,  $E(\text{edge})$  指的是图的边集。

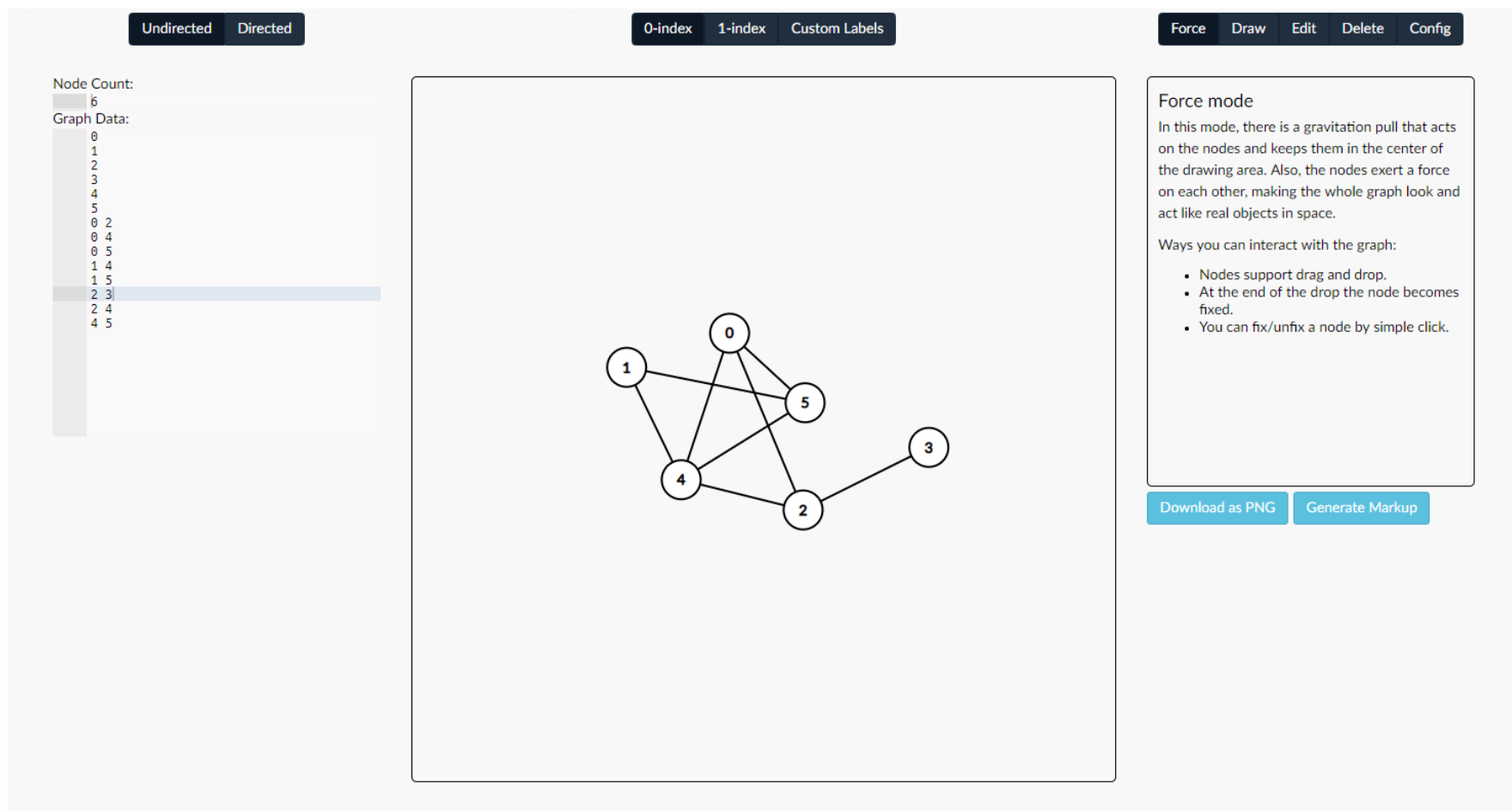
根据边是否有方向, 可将图分为有向图和无向图。

另外, 有些图的边上还可能有权值, 这样的图称为有权图。



## 在线做图

[https://csacademy.com/app/graph\\_editor/](https://csacademy.com/app/graph_editor/)



如果网站进不去就换个网络试试

## Matlab帮我们作图

% 函数`graph(s,t)`: 可在 `s` 和 `t` 中的对应节点之间创建边, 并生成一个图

```
G1 = graph(s1, t1);
```

```
plot(G1)
```

% 函数`graph(s,t,w)`: 可在 `s` 和 `t` 中的对应节点之间以 `w` 的权重创建边, 并生成一个图

```
G2 = graph(s2, t2);
```

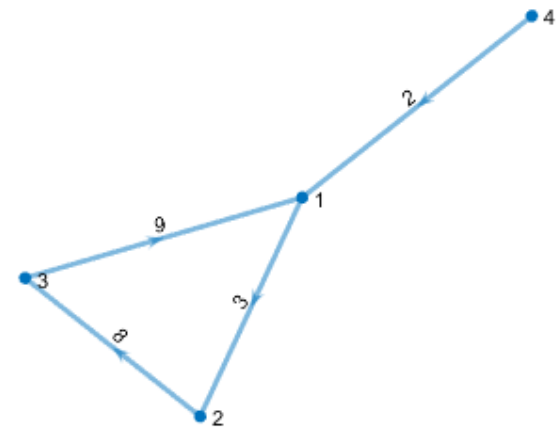
```
plot(G2, 'linewidth', 2) % 设置线的宽度
```

% 下面的命令是在画图后不显示坐标

```
set( gca, 'XTick', [], 'YTick', [] );
```

上面都是无向图

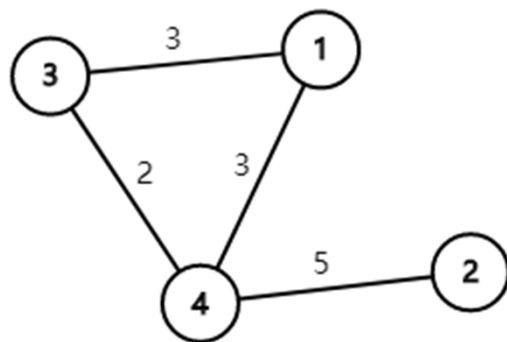
要做出有向图, 只需要将`graph`改为`digraph`就行了。



注: (1) Matlab做出来的图不是很漂亮, 要是节点比较少, 还是推荐大家使用在线作图。 **(2) 该函数在2015b之后的版本才支持, 如果运行出错请下载新版本Matlab。**

低版本Matlab报错提示: 未定义与 'double' 类型的输入参数相对应的函数 'graph'。

## 无向图的权重邻接矩阵



带权重的四个节点的无向图

$$D = \begin{bmatrix} 0 & Inf & 3 & 3 \\ Inf & 0 & Inf & 5 \\ 3 & Inf & 0 & 2 \\ 3 & 5 & 2 & 0 \end{bmatrix}$$

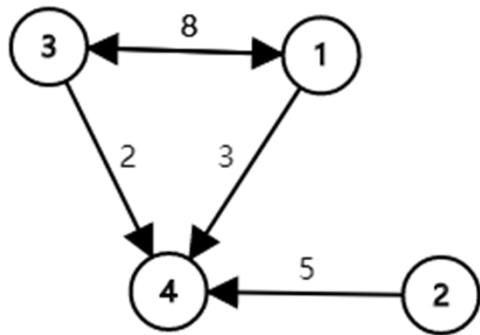
无向图对应的权重邻接矩阵

	1	2	3	4
1	0	Inf	3	3
2	Inf	0	Inf	5
3	3	Inf	0	2
4	3	5	2	0

结论:

- (1) 无向图对应的权重邻接矩阵D是一个对称矩阵;
- (2) 其主对角线上元素为0.
- (3)  $D_{ij}$ 表示第i个节点到第j个节点的权重。

## 有向图的权重邻接矩阵



带权重的四个节点的有向图

$$D = \begin{bmatrix} 0 & Inf & 8 & 3 \\ Inf & 0 & Inf & 5 \\ 8 & Inf & 0 & 2 \\ Inf & Inf & Inf & 0 \end{bmatrix}$$

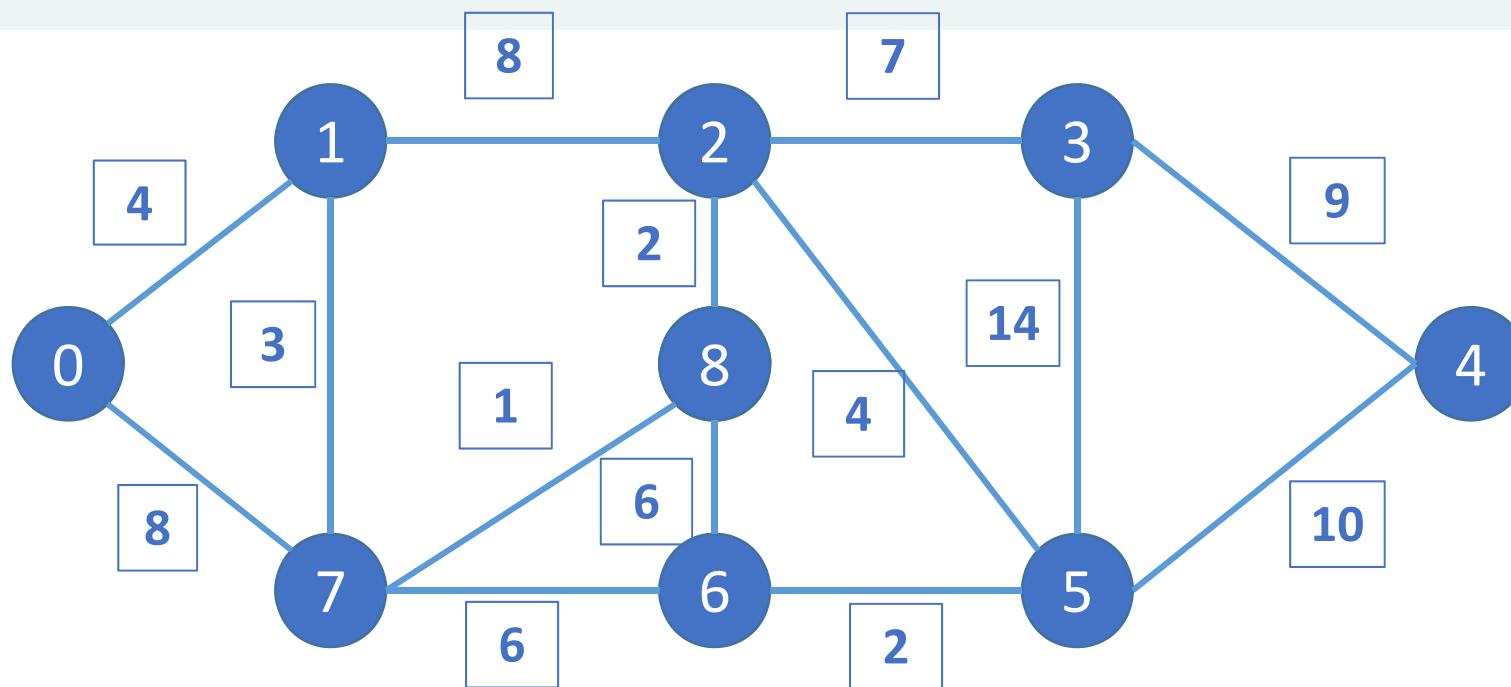
有向图对应的权重邻接矩阵

	1	2	3	4
1	0	Inf	8	3
2	Inf	0	Inf	5
3	8	Inf	0	2
4	Inf	Inf	Inf	0

结论:

- (1) 有向图对应的权重邻接矩阵D是一般不再是对称矩阵;
- (2) 其主对角线上元素为0.
- (3)  $D_{ij}$ 表示第i个节点到第j个节点的权重。

## 迪杰斯特拉算法



图中有0-8共九个地点, 地点之间若用直线连接则表明两地可直接到达, 直线旁的数值表示两地的距离。

问题: 起点为0, 终点为4, 怎么走路程最短。

(假设出行方式相同, 例如都为步行)



## 玩一个APP

### 算法动画图解 (安卓有破解版 苹果需要购买)

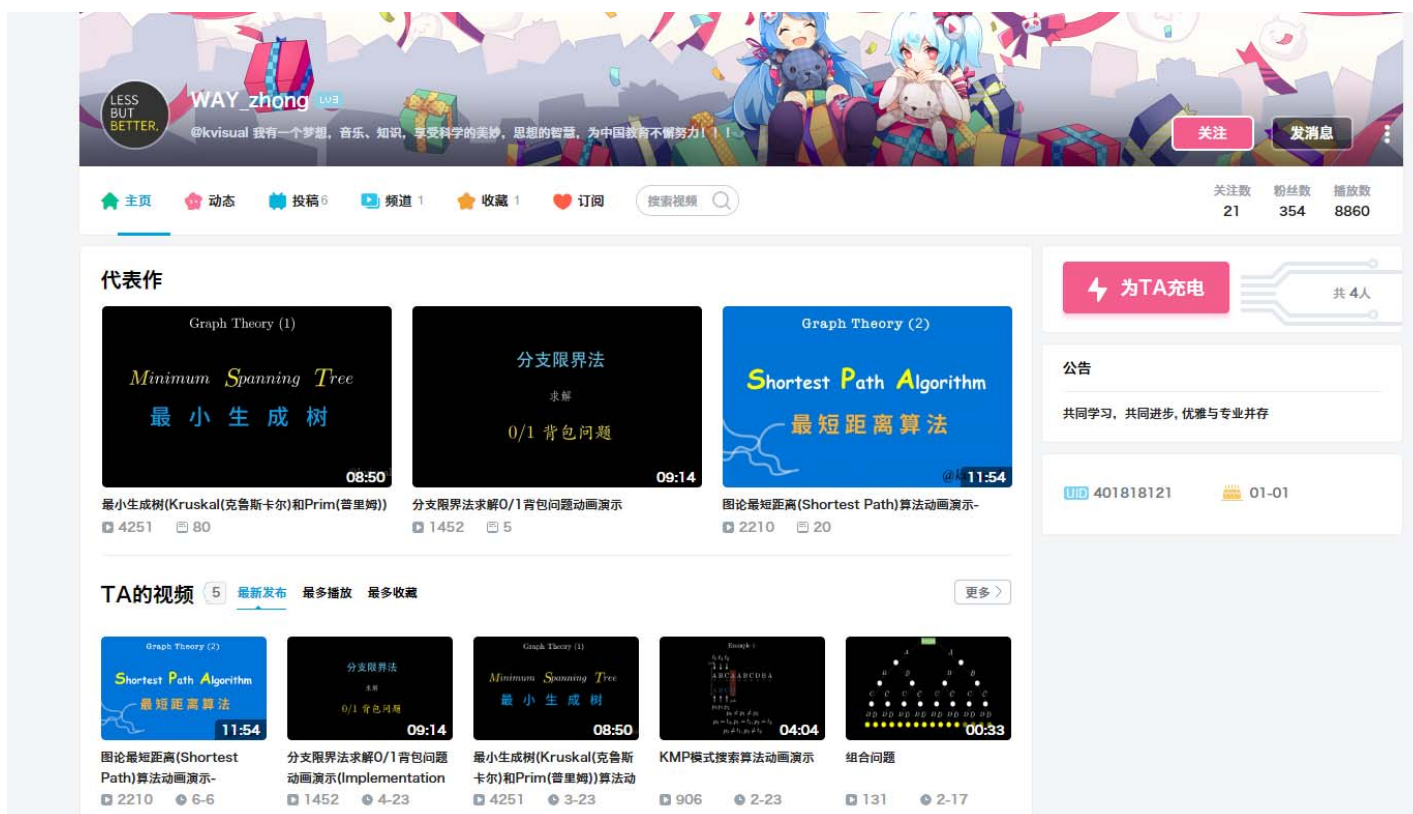
#### 图表搜索

-  广度优先搜索 
-  测试 
-  深度优先搜索 
-  测试 
-  贝尔曼-福特算法 
-  测试 
-  戴克斯特拉算法 
-  测试 
-  A\*搜索算法 
-  测试 

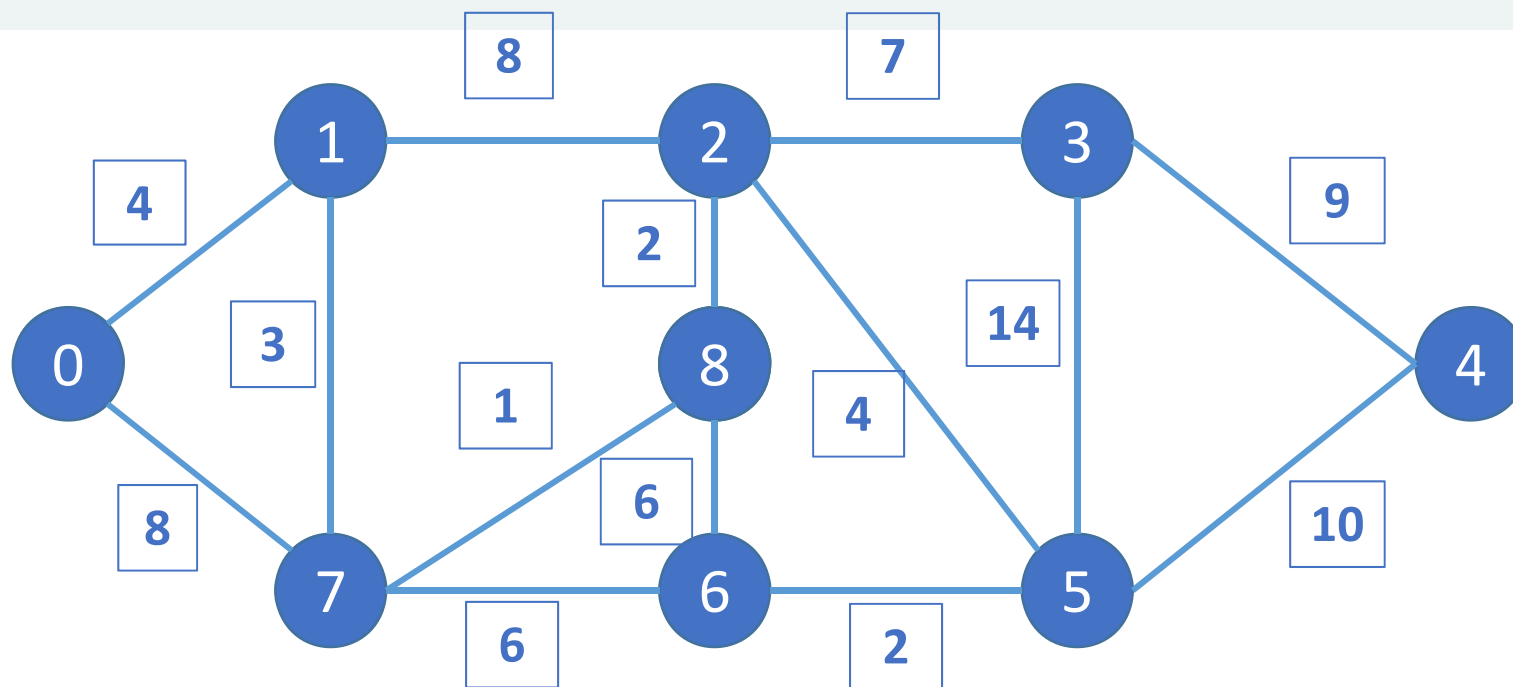
## 看视频演示

在线观看:

<https://www.bilibili.com/video/av54668527>

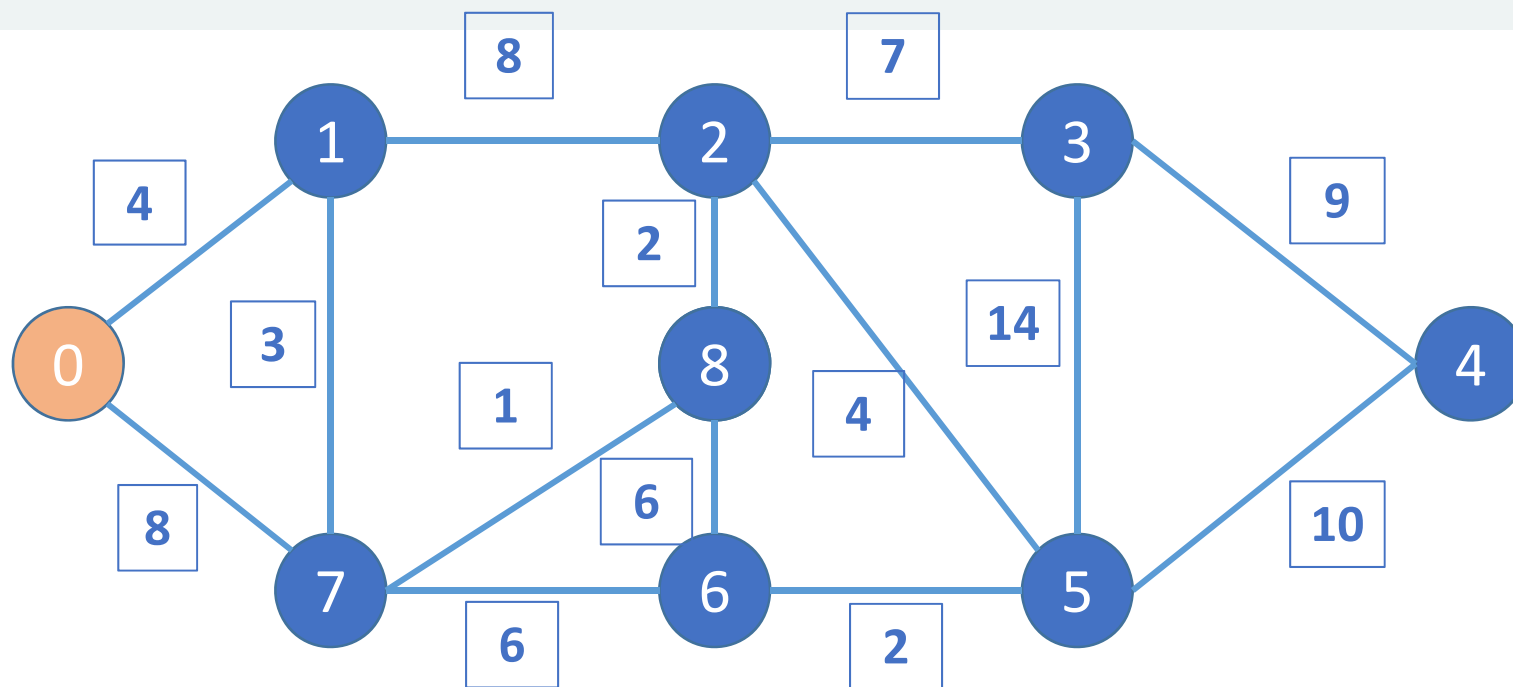


## 步骤演示



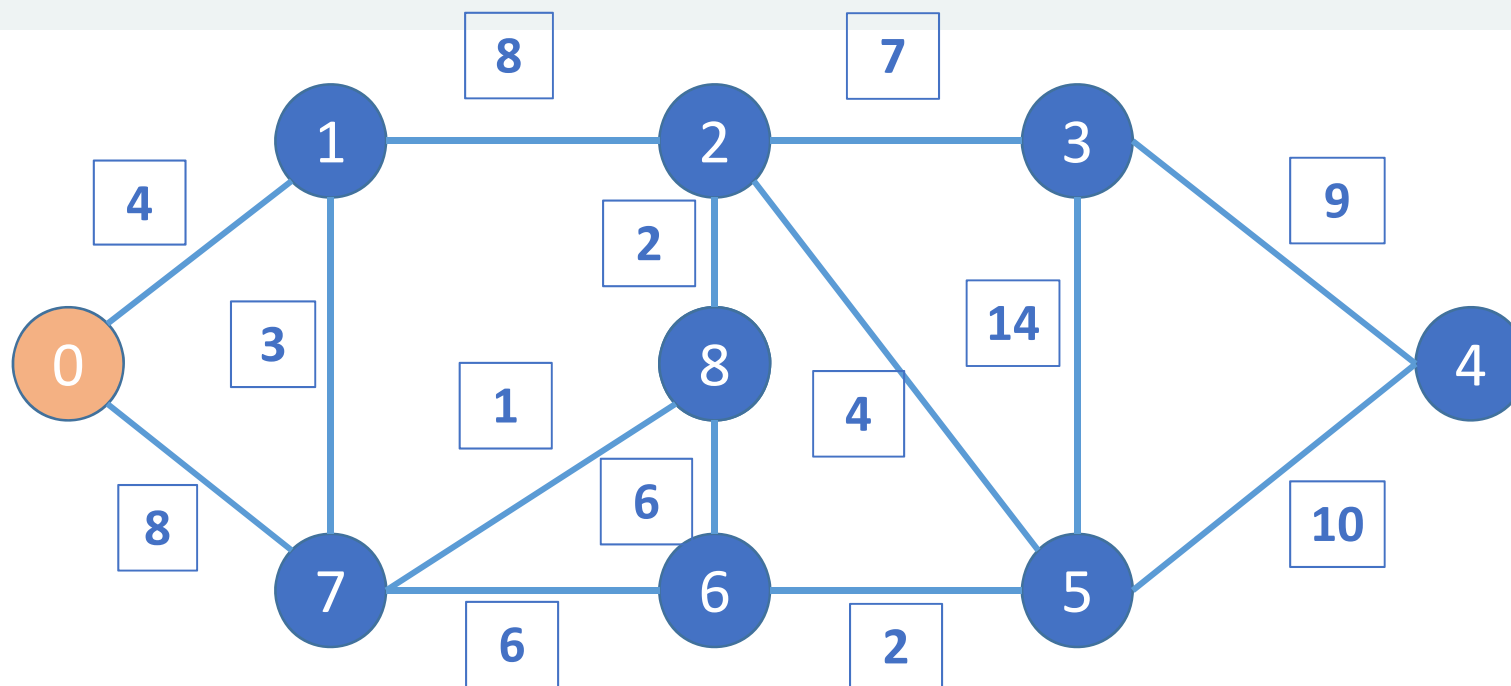
节点	0	1	2	3	4	5	6	7	8
Visited	0	0	0	0	0	0	0	0	0
Distance	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Parent	-1	-1	-1	-1	-1	-1	-1	-1	-1

## 步骤演示



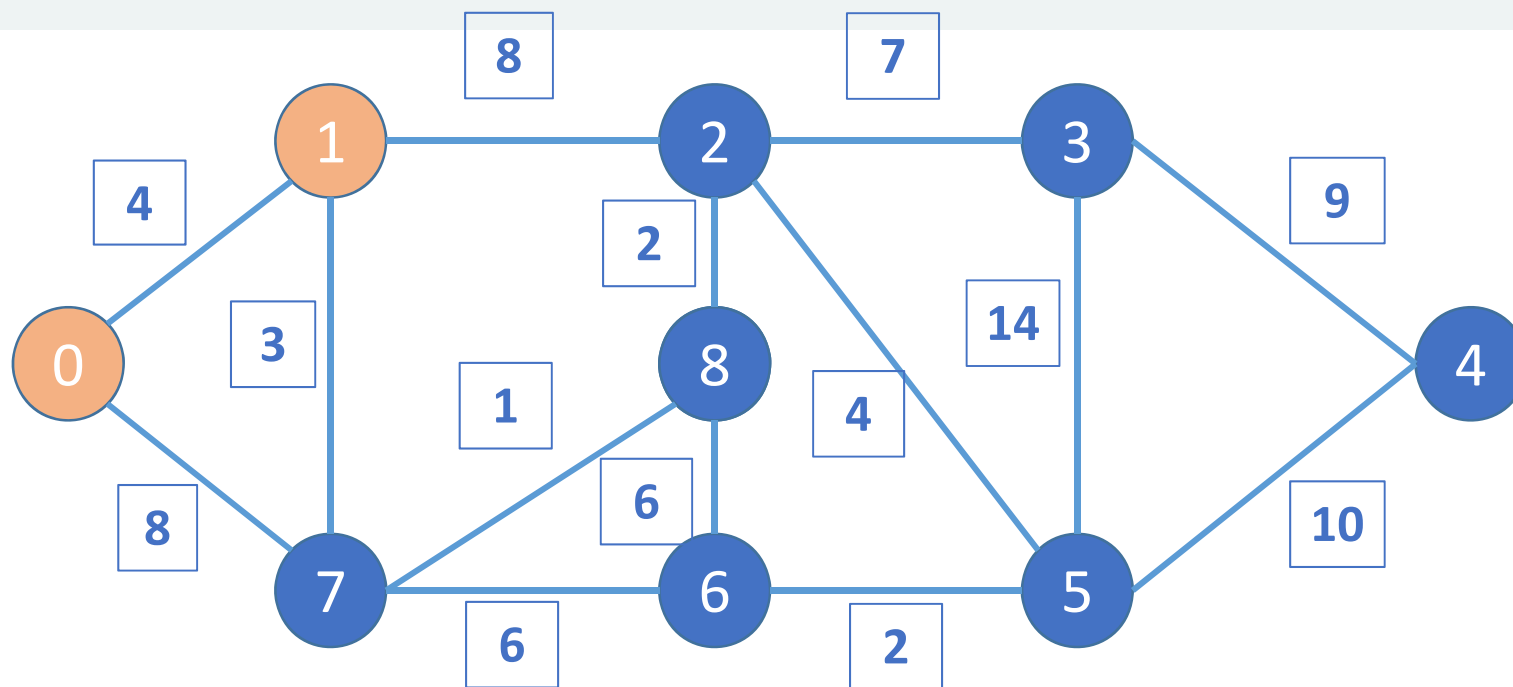
节点	0	1	2	3	4	5	6	7	8
Visited	1	0	0	0	0	0	0	0	0
Distance	0	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Parent	0	-1	-1	-1	-1	-1	-1	-1	-1

## 步骤演示



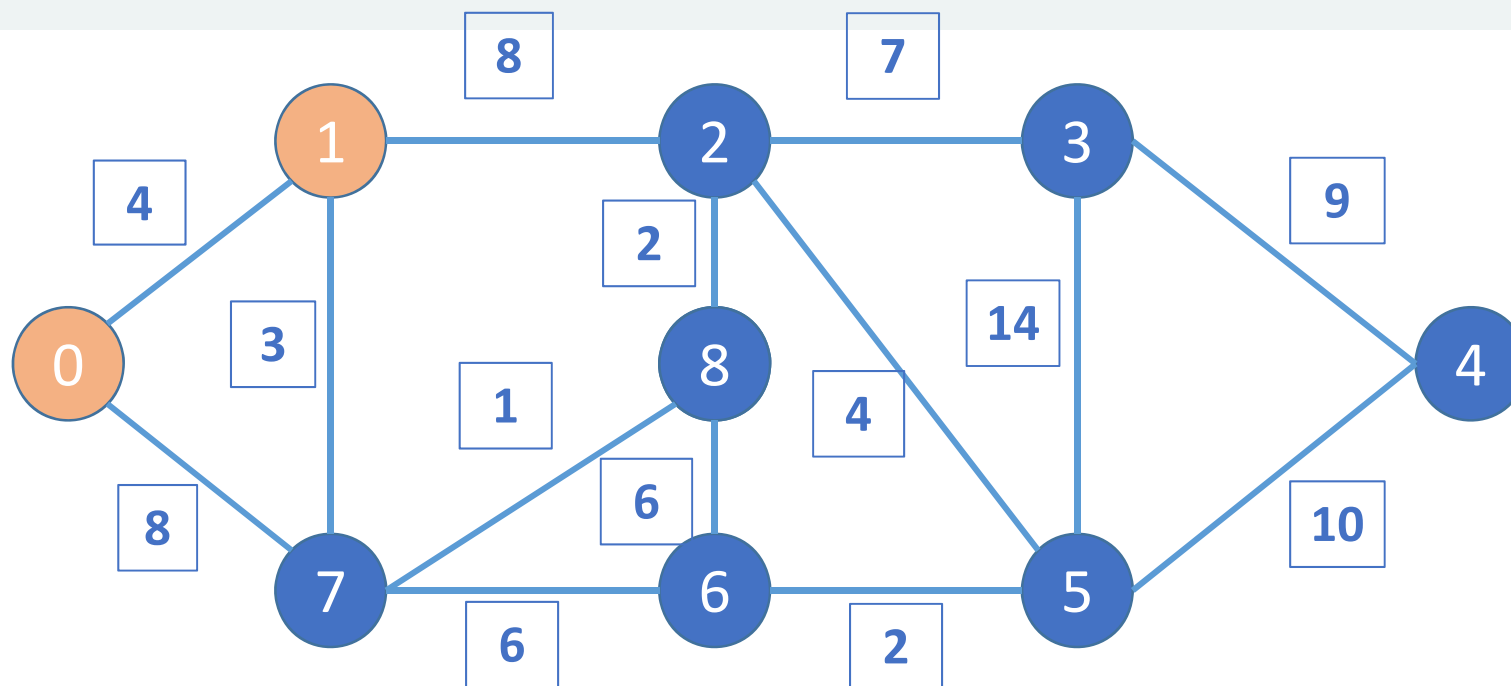
节点	0	1	2	3	4	5	6	7	8
Visited	1	0	0	0	0	0	0	0	0
Distance	0	4	Inf	Inf	Inf	Inf	Inf	8	Inf
Parent	0	0	-1	-1	-1	-1	-1	0	-1

## 步骤演示



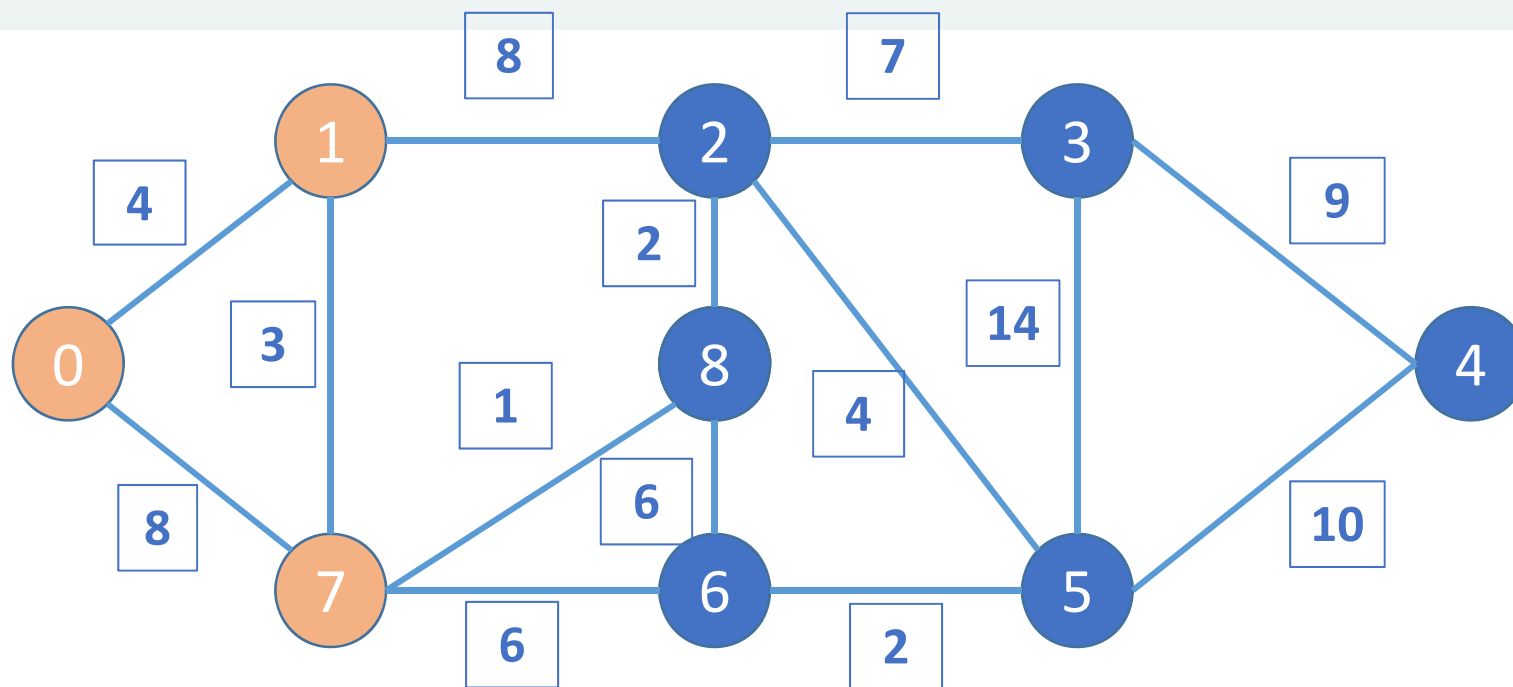
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	0	0
Distance	0	4	Inf	Inf	Inf	Inf	Inf	8	Inf
Parent	0	0	-1	-1	-1	-1	-1	0	-1

## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	0	0
Distance	0	4	12	Inf	Inf	Inf	Inf	7	Inf
Parent	0	0	1	-1	-1	-1	-1	1	-1

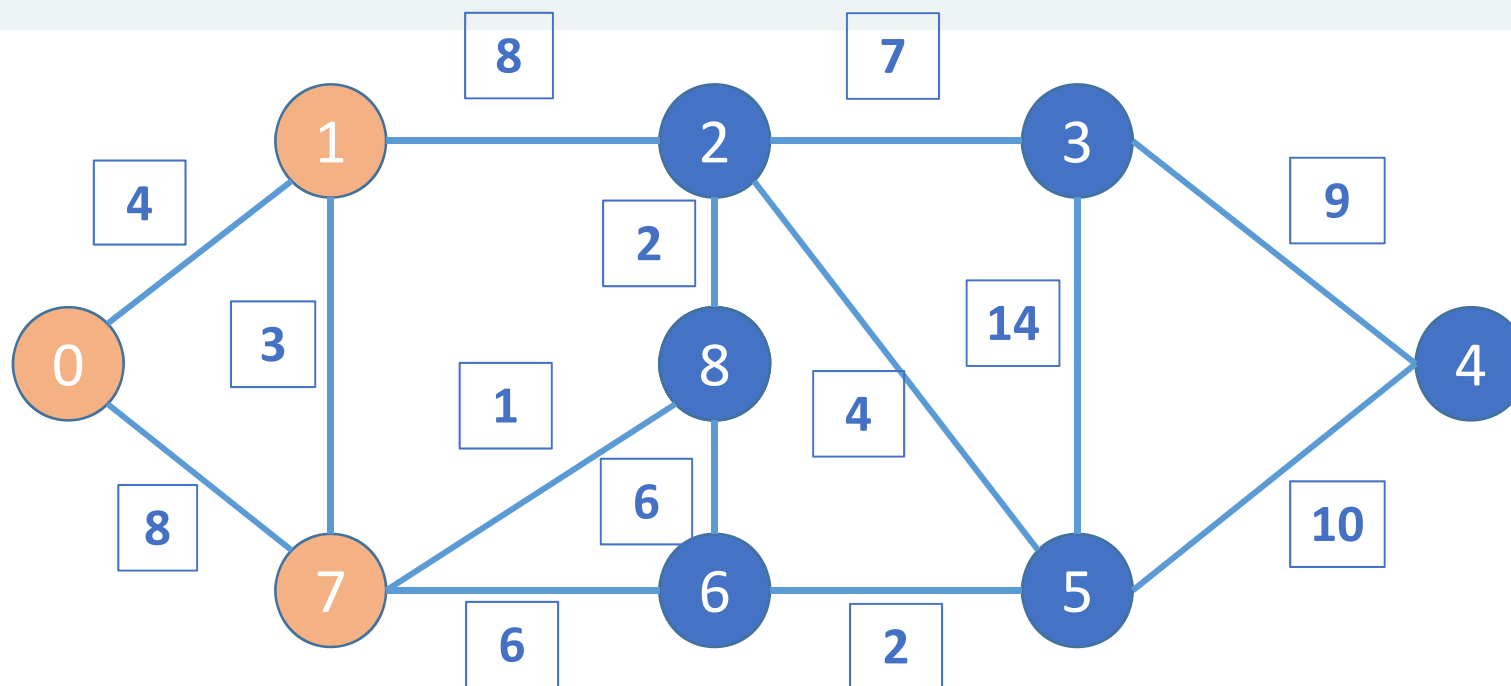
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	1	0
Distance	0	4	12	Inf	Inf	Inf	Inf	7	Inf
Parent	0	0	1	-1	-1	-1	-1	1	-1

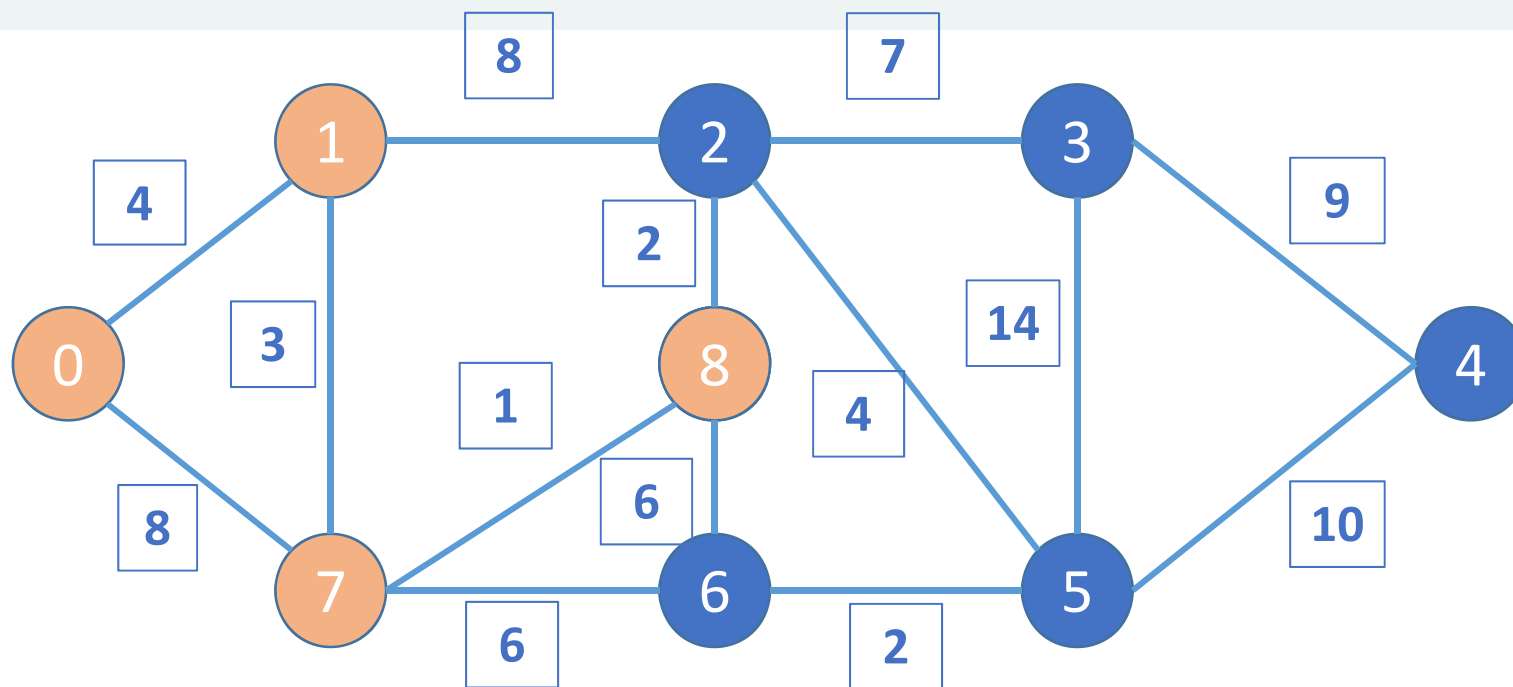


## 步骤演示



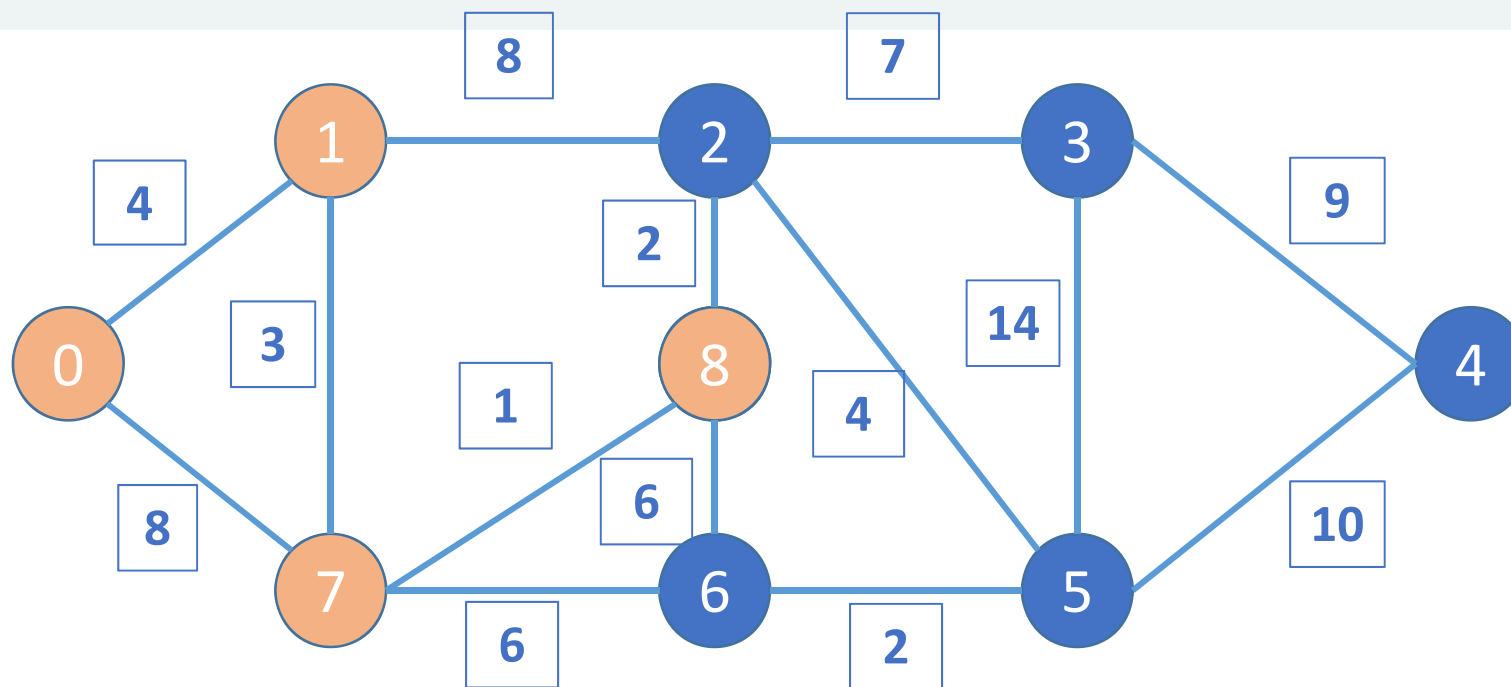
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	1	0
Distance	0	4	12	Inf	Inf	Inf	13	7	8
Parent	0	0	1	-1	-1	-1	7	1	7

## 步骤演示



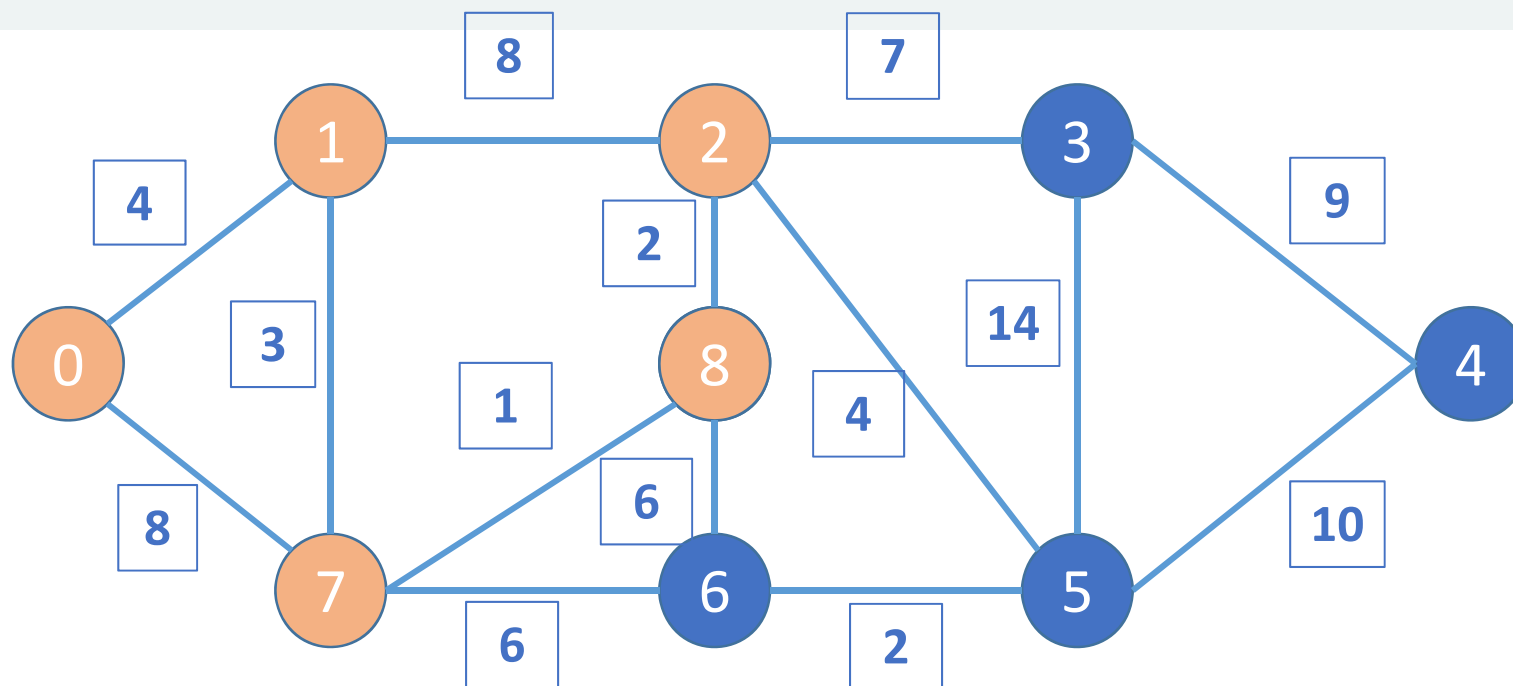
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	1	1
Distance	0	4	12	Inf	Inf	Inf	13	7	8
Parent	0	0	1	-1	-1	-1	7	1	7

## 步骤演示



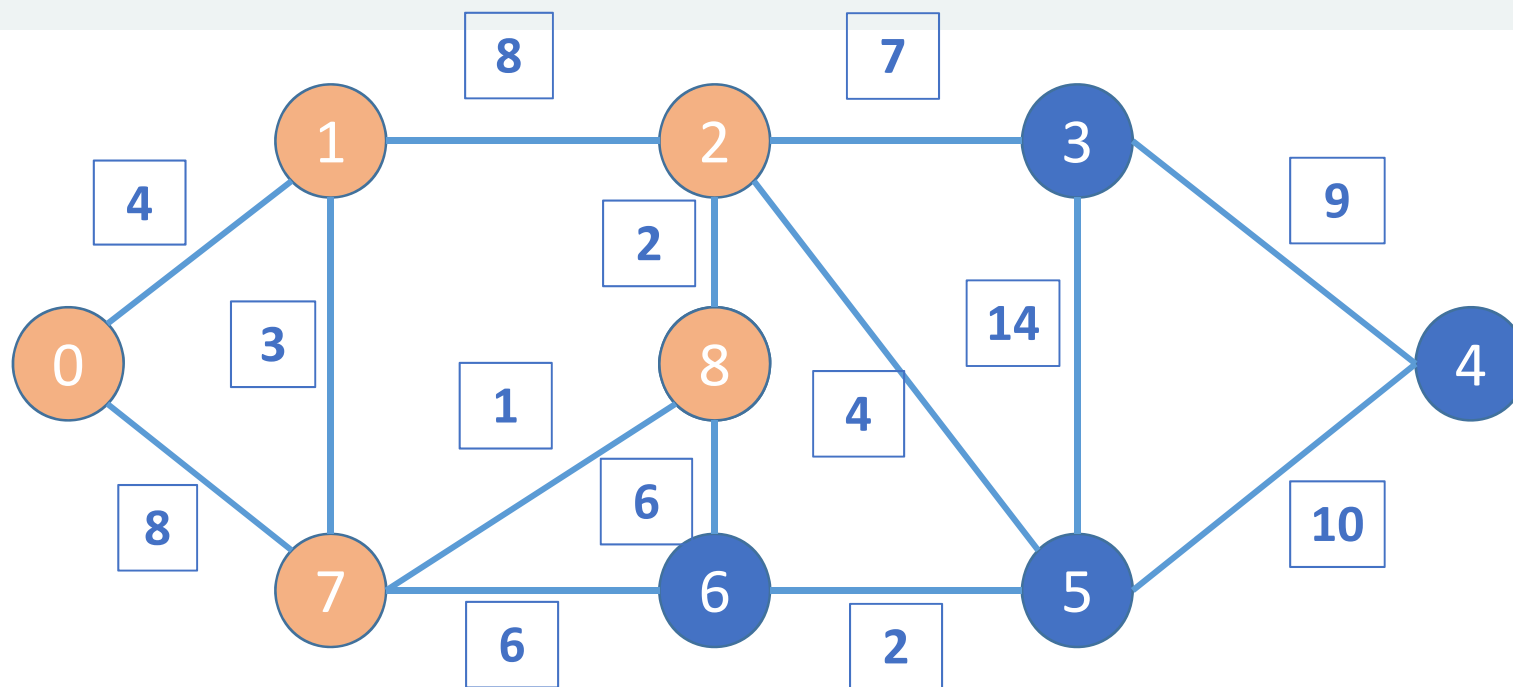
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	0	0	0	0	0	1	1
Distance	0	4	10	Inf	Inf	Inf	13	7	8
Parent	0	0	8	-1	-1	-1	7	1	7

## 步骤演示



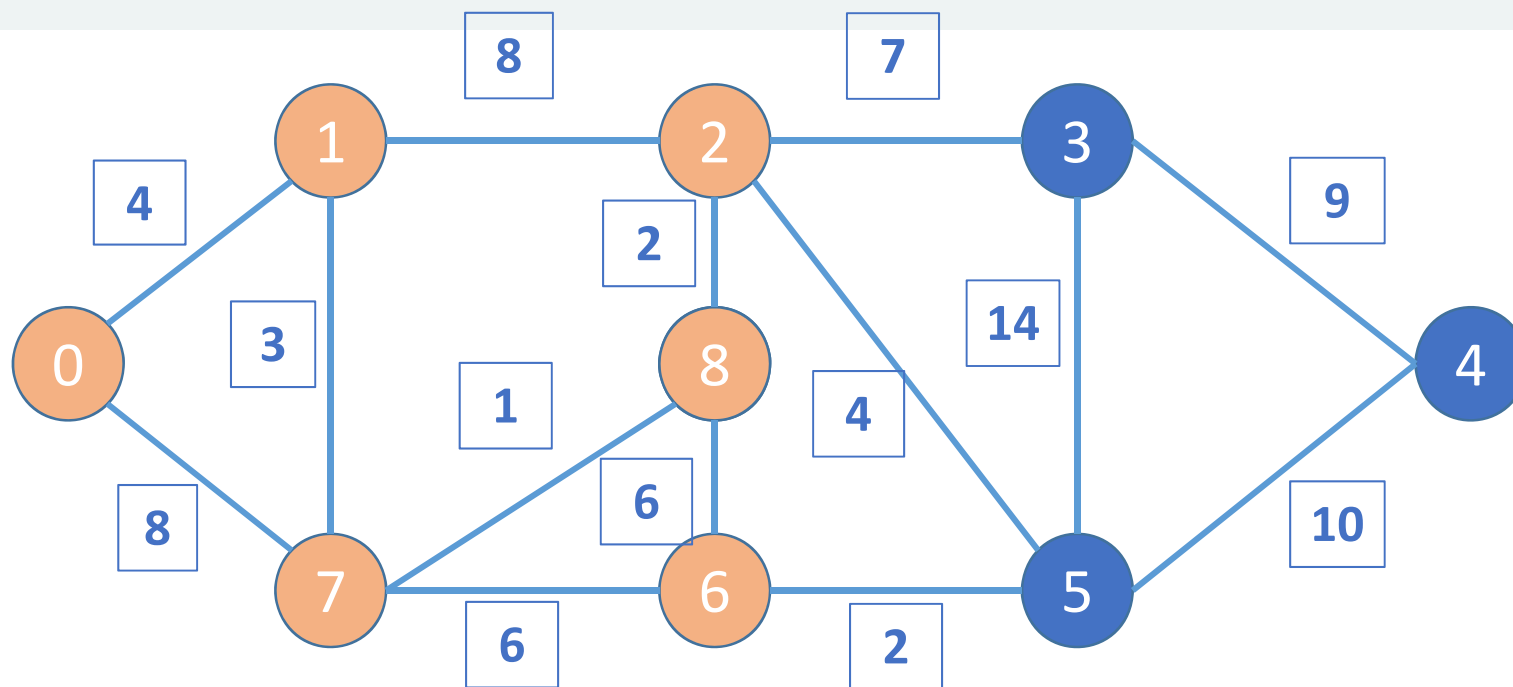
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	0	0	1	1
Distance	0	4	10	Inf	Inf	Inf	13	7	8
Parent	0	0	8	-1	-1	-1	7	1	7

## 步骤演示



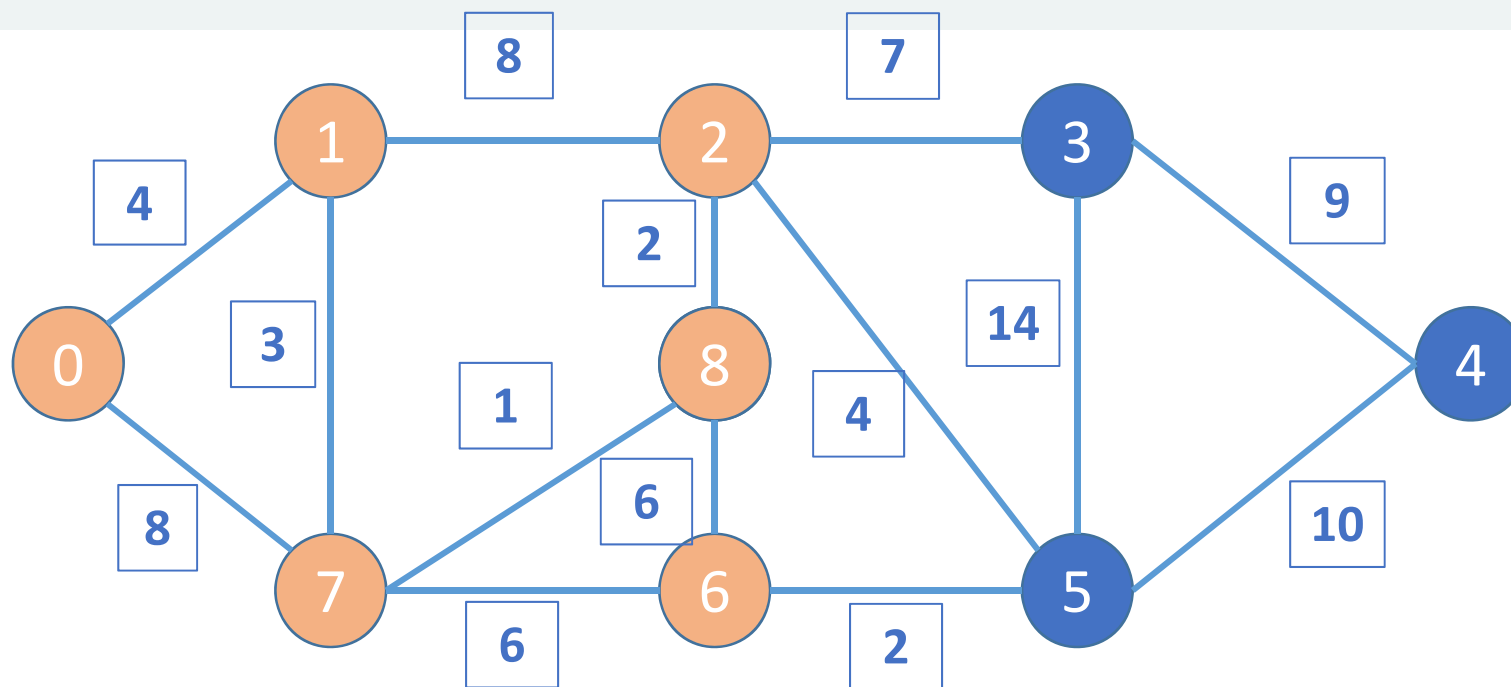
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	0	0	1	1
Distance	0	4	10	17	Inf	14	13	7	8
Parent	0	0	8	2	-1	2	7	1	7

## 步骤演示



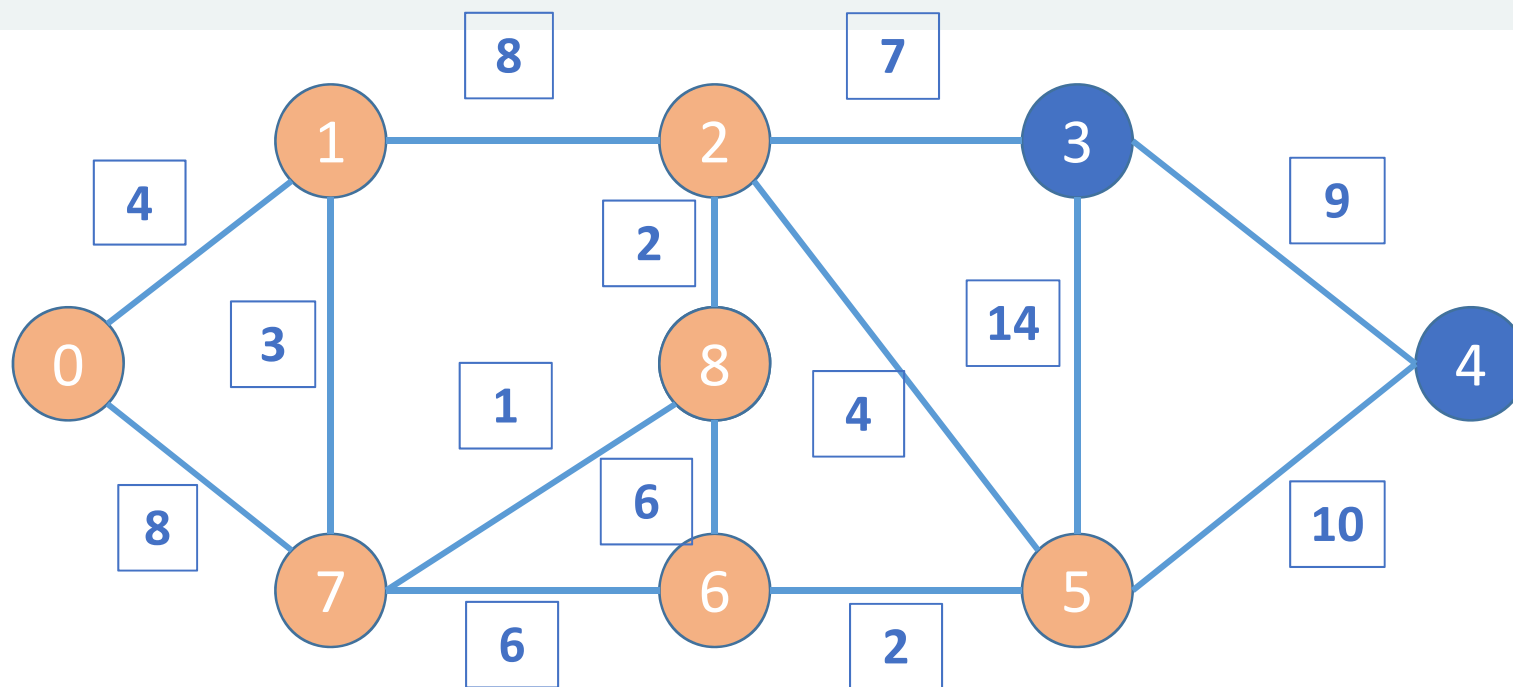
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	0	1	1	1
Distance	0	4	10	17	Inf	14	13	7	8
Parent	0	0	8	2	-1	2	7	1	7

## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	0	1	1	1
Distance	0	4	10	17	Inf	14	13	7	8
Parent	0	0	8	2	-1	2	7	1	7

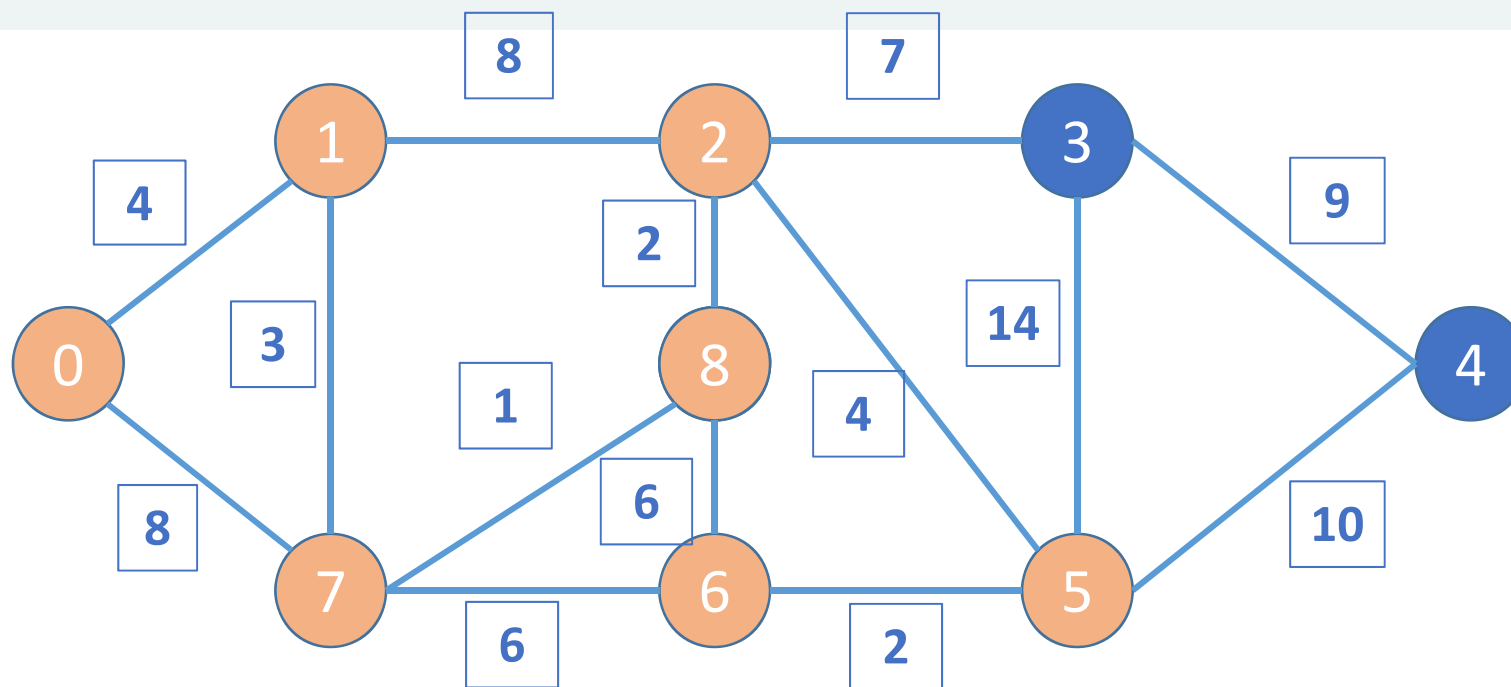
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	1	1	1	1
Distance	0	4	10	17	Inf	14	13	7	8
Parent	0	0	8	2	-1	2	7	1	7

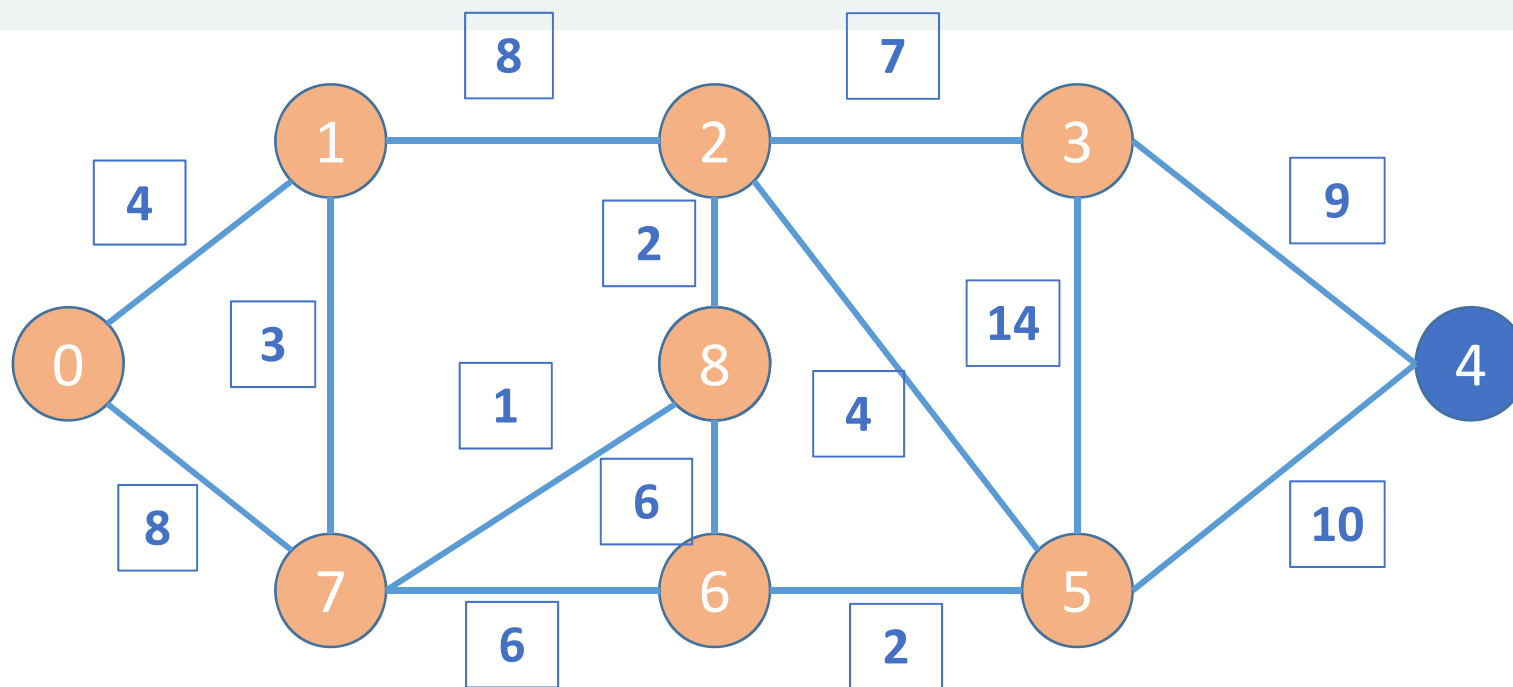


## 步骤演示



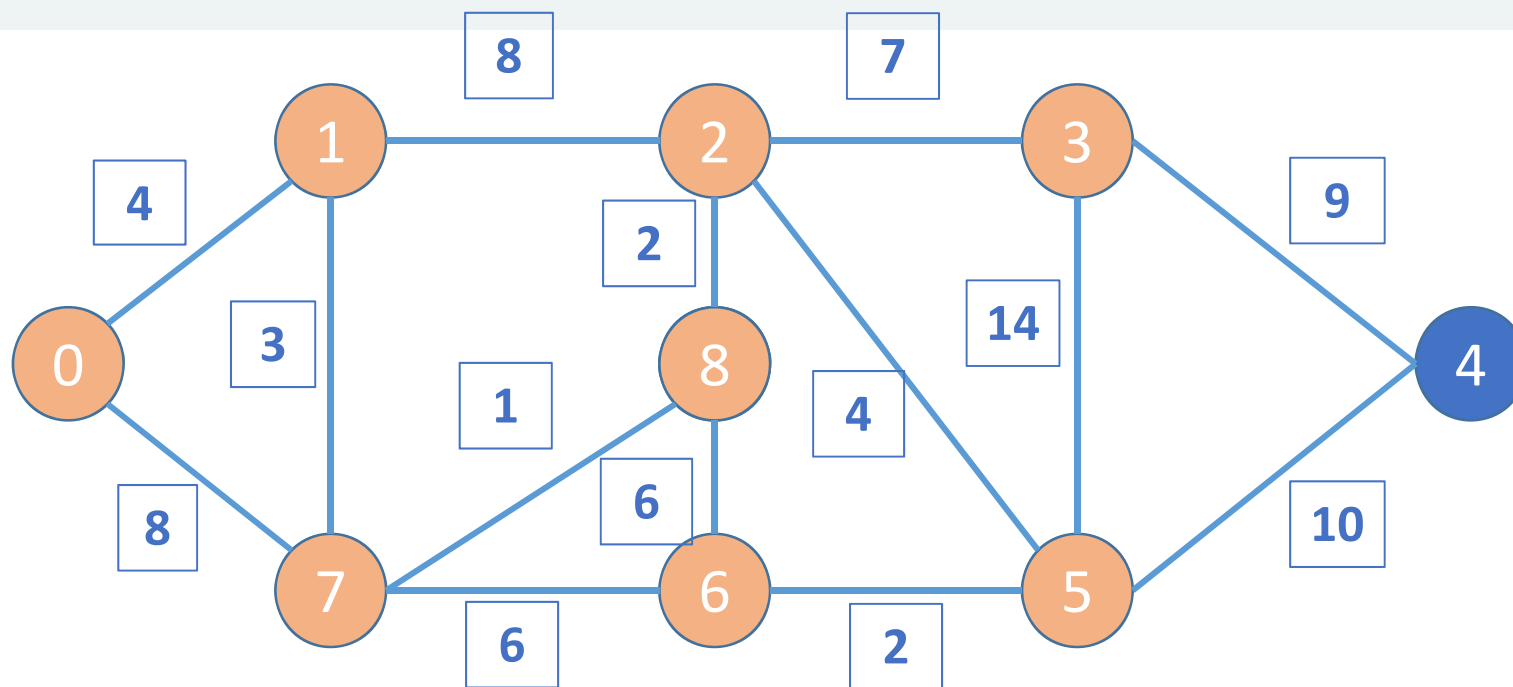
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	0	0	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

## 步骤演示



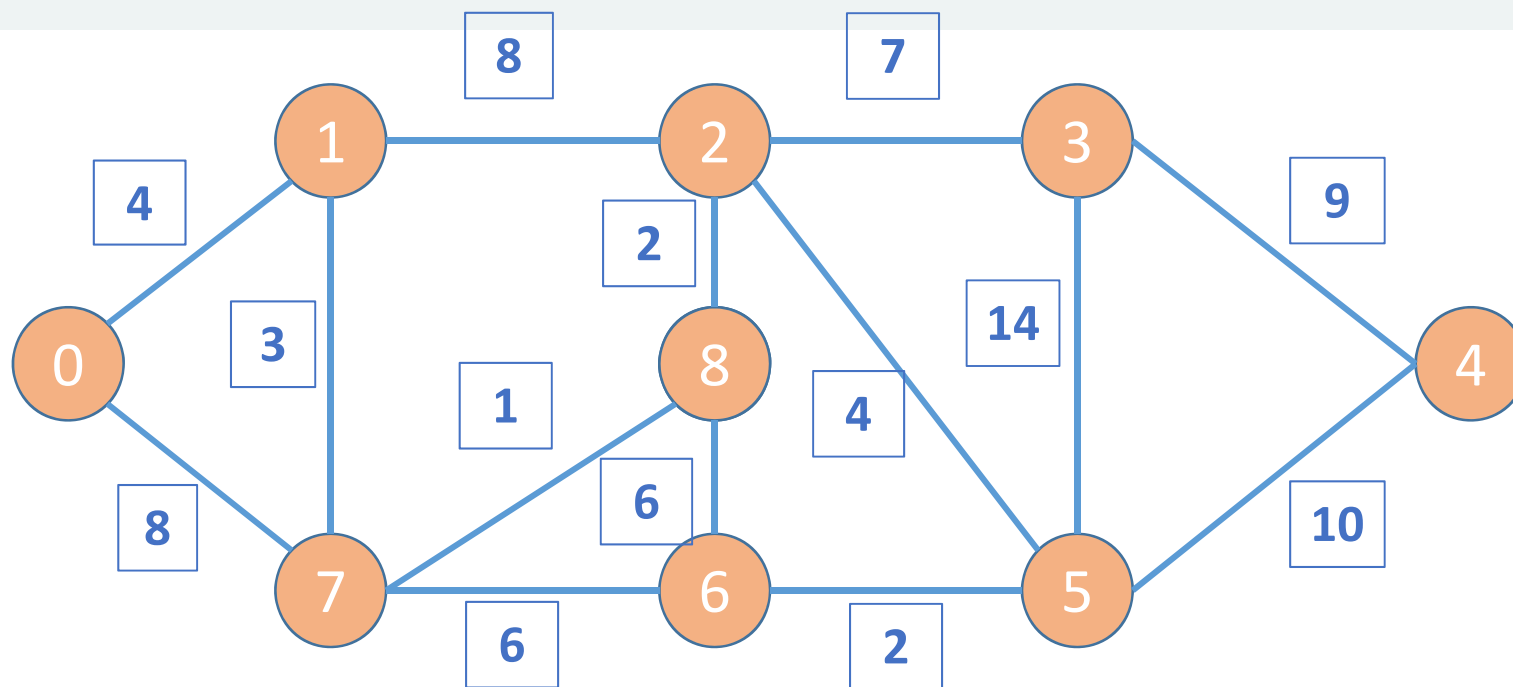
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	0	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

## 步骤演示



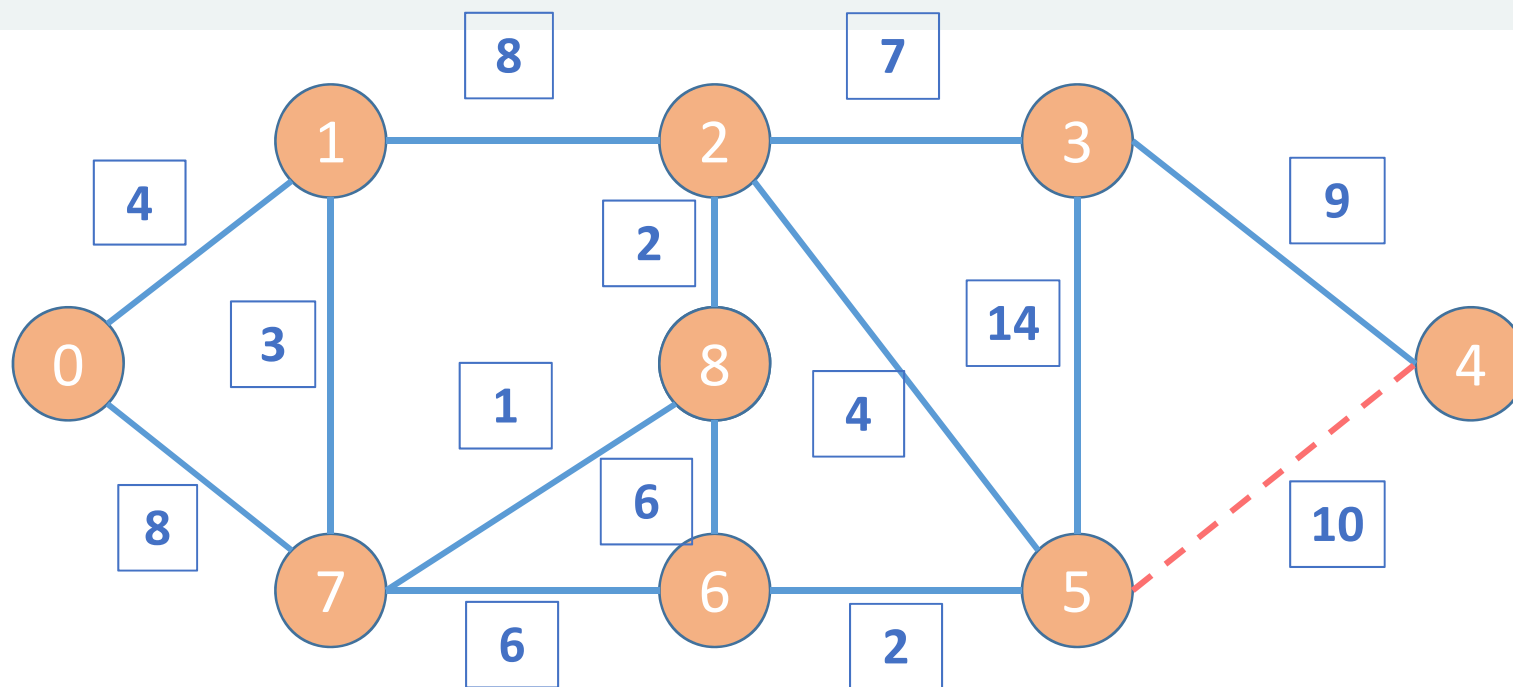
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	0	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

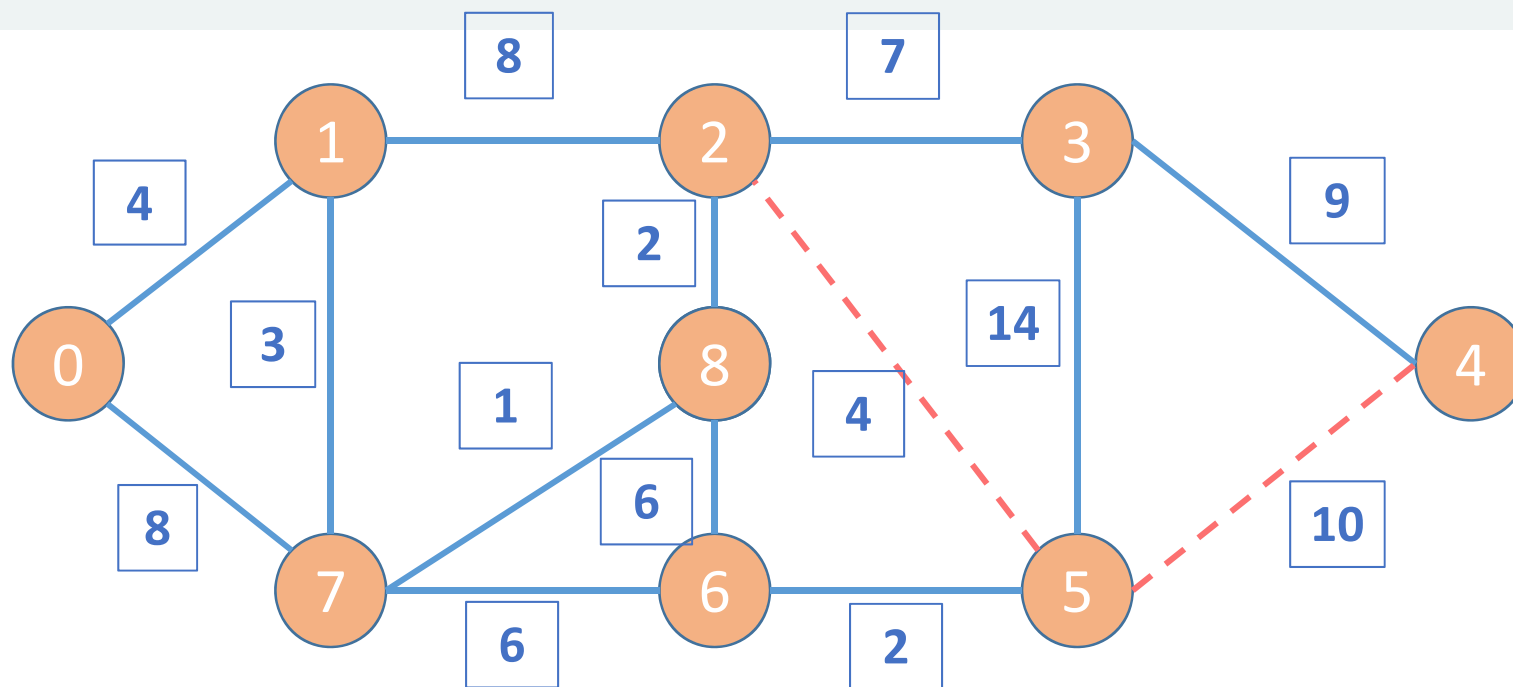
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

 $4 \leftarrow 5$

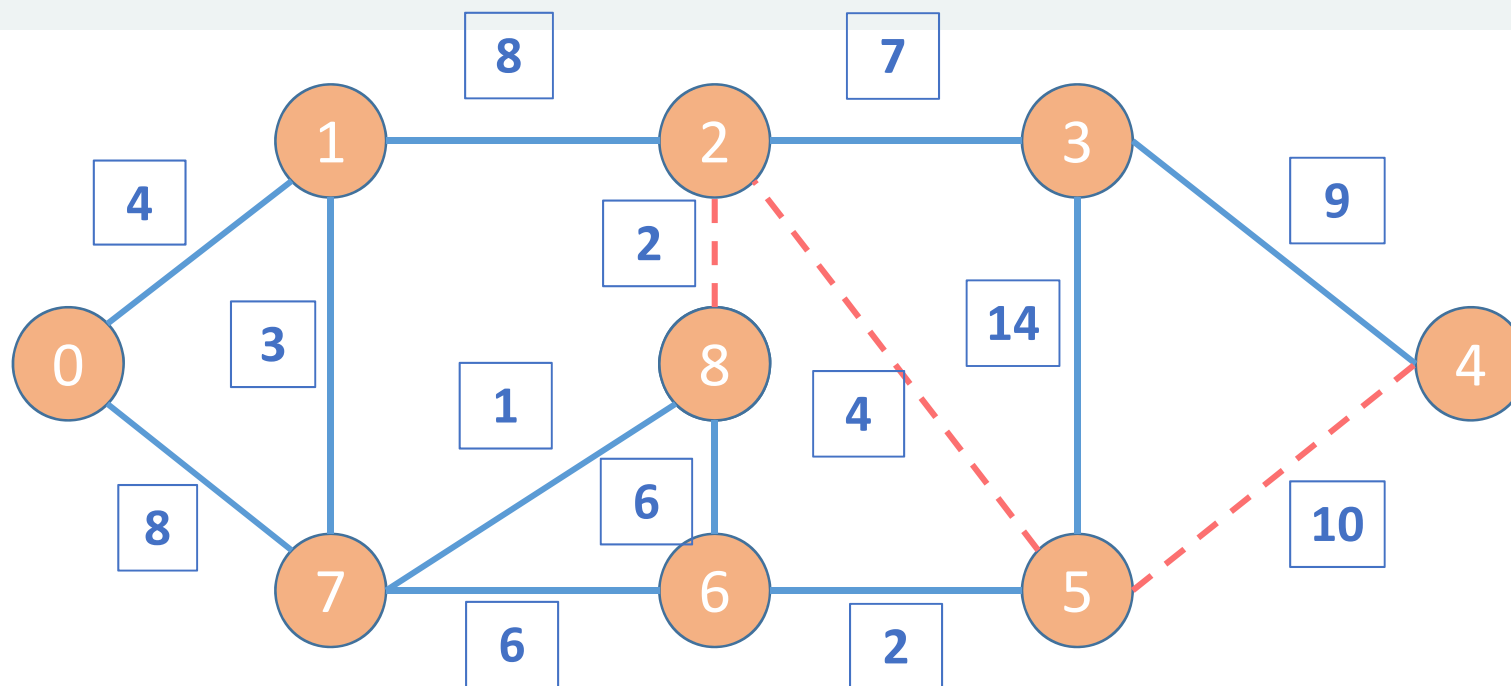
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

$$4 \leftarrow 5 \leftarrow 2$$

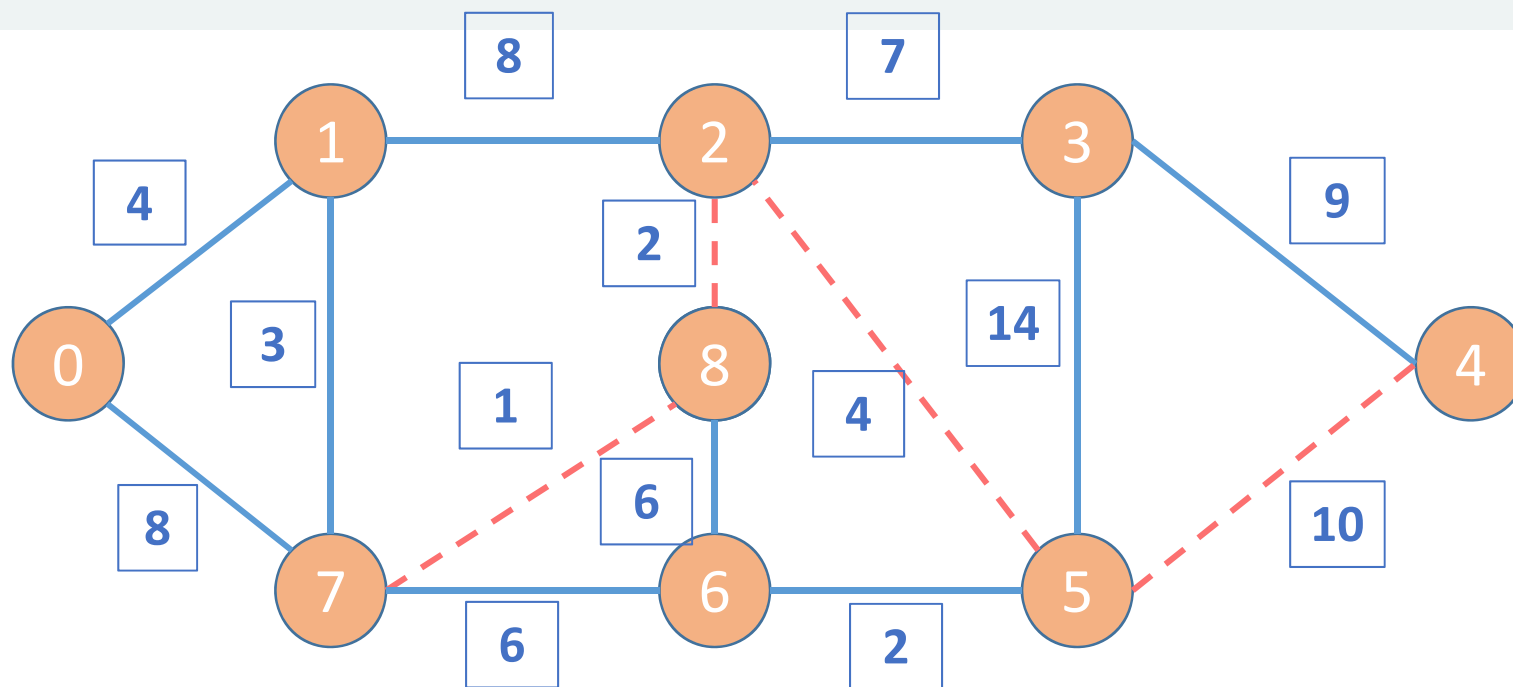
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

$4 \leftarrow 5 \leftarrow 2 \leftarrow 8$

## 步骤演示

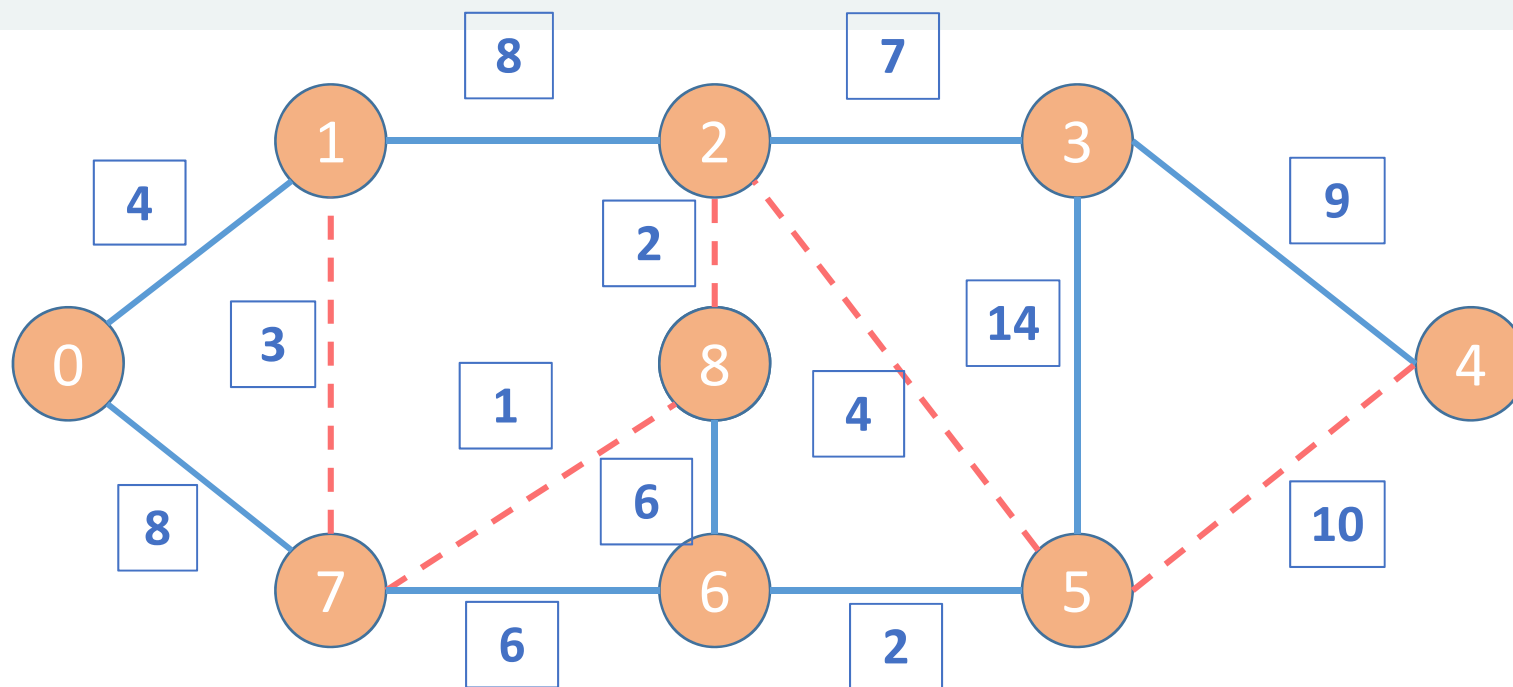


节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

$$4 \leftarrow 5 \leftarrow 2 \leftarrow 8 \leftarrow 7$$



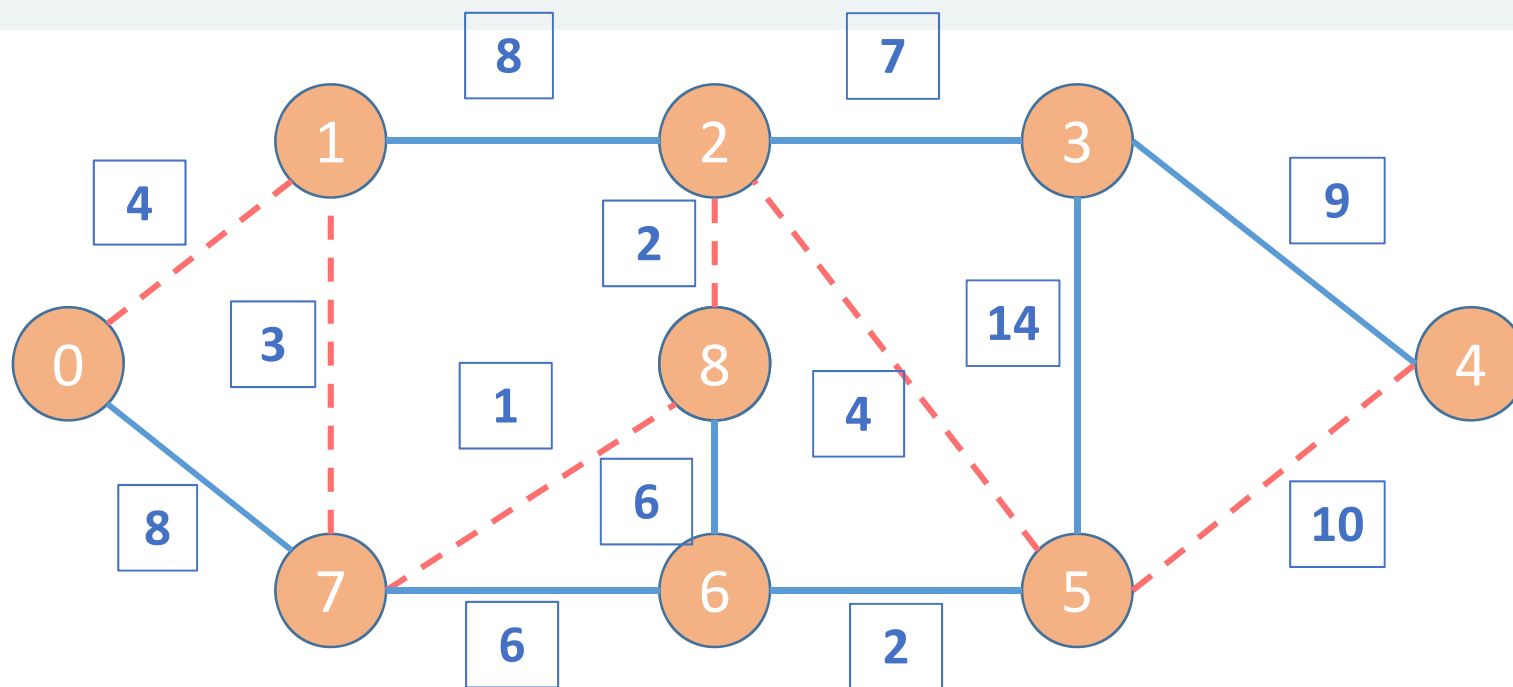
## 步骤演示



节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

$$4 \leftarrow 5 \leftarrow 2 \leftarrow 8 \leftarrow 7 \leftarrow 1$$

## 步骤演示



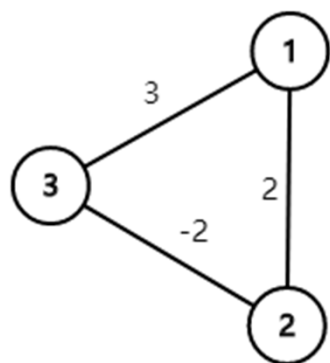
节点	0	1	2	3	4	5	6	7	8
Visited	1	1	1	1	1	1	1	1	1
Distance	0	4	10	17	24	14	13	7	8
Parent	0	0	8	2	5	2	7	1	7

$$4 \leftarrow 5 \leftarrow 2 \leftarrow 8 \leftarrow 7 \leftarrow 1 \leftarrow 0$$

# 迪杰斯特拉算法的一个缺点

可以用于有向图

但不能处理负权重



1是起点; 2是终点

节点	1	2	3
Visited	0	0	0
Distance	Inf	Inf	Inf
Parent	-1	-1	-1

(1)

节点	1	2	3
Visited	1	0	0
Distance	0	2	3
Parent	1	1	1

(3)

节点	1	2	3
Visited	1	1	0
Distance	0	2	0
Parent	1	1	2

(5)

节点	1	2	3
Visited	1	0	0
Distance	0	Inf	Inf
Parent	1	-1	-1

(2)

节点	1	2	3
Visited	1	1	0
Distance	0	2	3
Parent	1	1	1

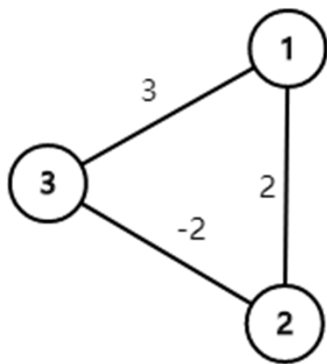
(4)

节点	1	2	3
Visited	1	1	1
Distance	0	2	0
Parent	1	1	2

(6)

## 如何修复该缺点?

### Bellman-Ford (贝尔曼-福特) 算法



1是起点; 2是终点

刚刚改变访问状态的节点为0号节点 (A)

我们要更新与0号节点相邻的节点信息 (B), 注意, 这里的B节点是未访问的哦

更新的规则如下:

如果 (A与B的距离+ A列表中的距离) 小于 (B列表中的距离), 那么我们就将B列表中的距离更新为较小的距离, 并将B的父亲节点更新为A

**事实上, 贝尔曼-福特算法不再将节点区分为是否已访问的状态, 因为贝尔曼-福特模型是利用循环来进行更新权重的, 且每循环一次, 贝尔曼福特算法都会更新所有的节点的信息。**

贝尔曼-福特算法不支持含有负权回路的图。

(视频中提到的Floyd(弗洛伊德)算法也不可以)

有兴趣的同学可以参考下面两份资料弄懂其实现原理:

<https://blog.csdn.net/a8082649/article/details/81812000>

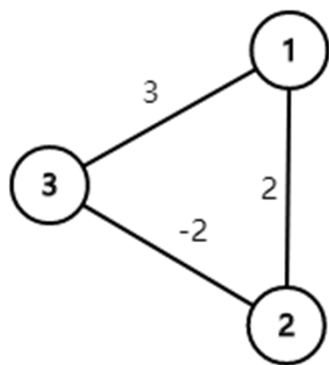
<https://www.bilibili.com/video/av43217121>

## 什么是负权回路?

在一个图里每条边都有一个权值 (有正有负)

如果存在一个环 (从某个点出发又回到自己的路径), 而且这个环上所有权值之和是负数, 那这就是一个负权环, 也叫负权回路。

存在负权回路的图是不能求两点间最短路的, 因为只要在负权回路上不断兜圈子, 所得的最短路长度可以任意小。



含有负权重的无向图都是负权回路。  
例如左图, 可以在2-3之间无限循环。

**注意:**

贝尔曼-福特算法实际上处理的是具有负权重的**有向图**。  
(且该有向图也不能含有负权回路)

庆幸的是, 含有负权重的图特别少见, 且一旦出现负权重, 也往往是在有向图中。  
因此大家不用担心算法求解不出来的问题。

## Matlab计算最短路径

```
[P,d] = shortestpath(G,start,end [, 'Method',algorithm] )
```

**功能：** 返回图G中start节点到end节点的最短路径

**输入参数：**

- (1) G - 输入图 (graph 对象 | digraph 对象)
- (2) start 起始的节点
- (3) end 目标的节点
- (4) ['Method',algorithm]是可选的参数, 表示计算最短路径的算法。一般我们不用手动设置, 默认使用的是“auto”, 具体可设置的参数见下一页课件。

**输出参数：**

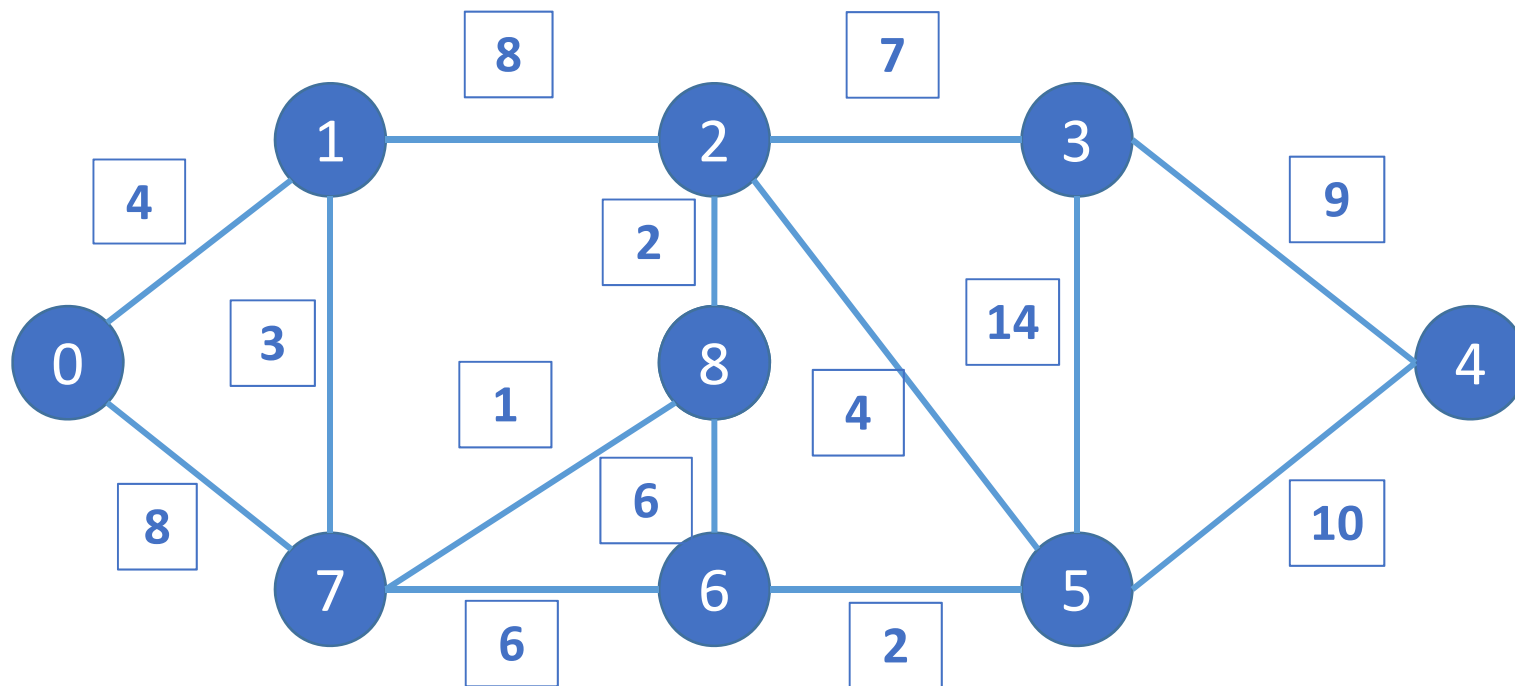
- (1) P - 最短路径经过的节点
- (2) d - 最短距离

注意：该函数matlab2015b之后才有哦

## 可选的算法

选项	说明
'auto' (默认值)	<p>'auto' 选项会自动选择算法:</p> <ul style="list-style-type: none"> <li>• 'unweighted' 用于没有边权重的 graph 和 digraph 输入。</li> <li>• 'positive' 用于具有边权重的所有 graph 输入, 并要求权重为非负数。此选项还用于具有非负边权重的 digraph 输入。</li> <li>• 'mixed' 用于其边权重包含某些负值的 digraph 输入。图不能包含负循环。</li> </ul>
'unweighted'	广度优先计算, 将所有边权重都视为 1。
'positive'	Dijkstra 算法, 要求所有边权重均为非负数。
'mixed' (仅适用于 digraph)	适用于有向图的 Bellman-Ford 算法, 要求图没有负循环。尽管对于相同的问题, 'mixed' 的速度慢于 'positive', 但 'mixed' 更为通用, 因为它允许某些边权重为负数。

## Matlab演示



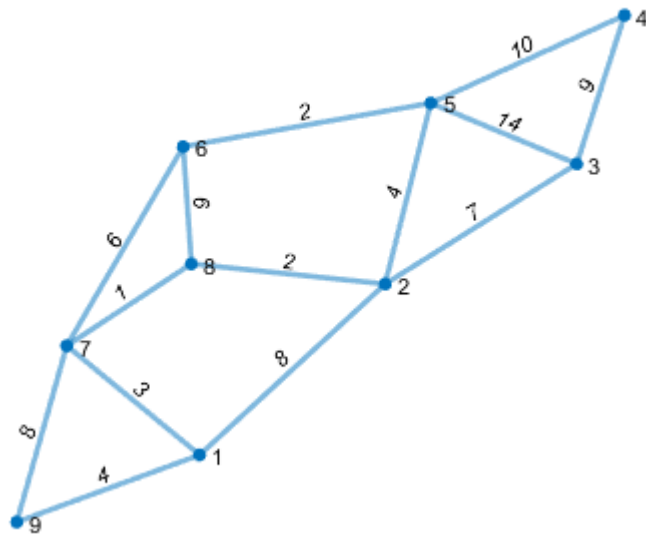
% 注意哦, Matlab中的图节点要从1开始编号, 所以这里把0全部改为了9

% 编号最好是从1开始连续编号, 不要自己随便定义编号



## 返回任意两点的距离矩阵

```
d = distances(G [, 'Method', algorithm])
```



```
>> D = distances(G)
```

D =

0	6	13	20	10	9	3	4	4
6	0	7	14	4	6	3	2	10
13	7	0	9	11	13	10	9	17
20	14	9	0	10	12	17	16	24
10	4	11	10	0	2	7	6	14
9	6	13	12	2	0	6	6	13
3	3	10	17	7	6	0	1	7
4	2	9	16	6	6	1	0	8
4	10	17	24	14	13	7	8	0

注意：该函数matlab2015b之后才有哦

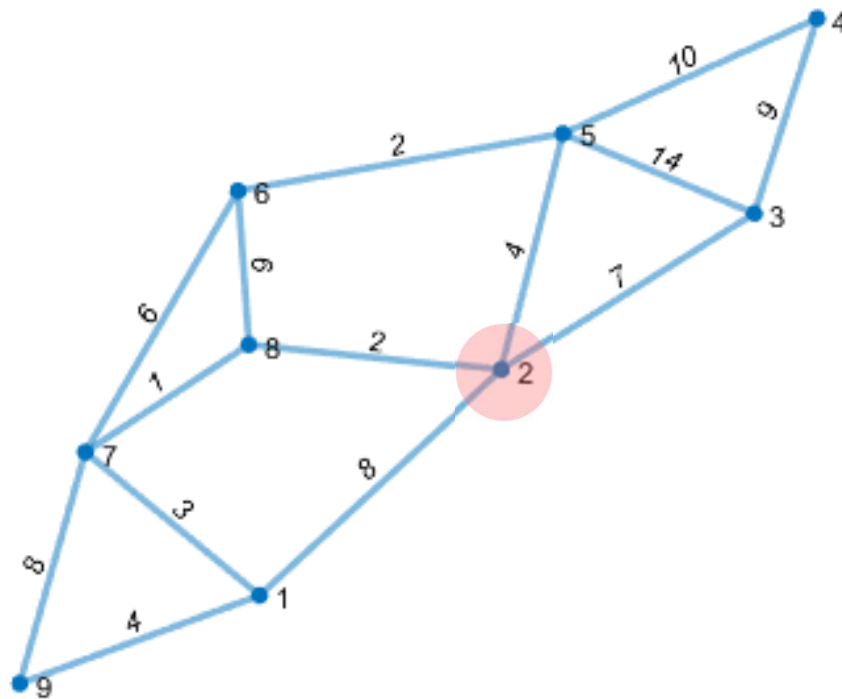
## 找给定范围内所有的点

```
[nodeIDs,dist] = nearest(G,s,d [, 'Method',algorithm])
```

返回图形 G 中与节点 s 的距离在 d 之内的所有节点。

nodeIDs是符合条件的节点

Dist是这些节点与s的距离



```
>> [nodeIDs,dist] = nearest(G, 2, 10)
```

```
nodeIDs =
```

```
8
7
5
1
6
3
9
```

```
dist =
```

```
2
3
4
6
6
7
10
```

注意: 该函数matlab2016a之后才有哦

## 课后作业

### 代码作业 (选做) :

自己实现迪杰斯特拉算法, 可定义一个函数。

`[P,d] = Dijkstra(D,start,end)`

#### 输入参数:

- (1) D是权重邻接矩阵
- (2) start 起始的节点
- (3) end 目标的节点

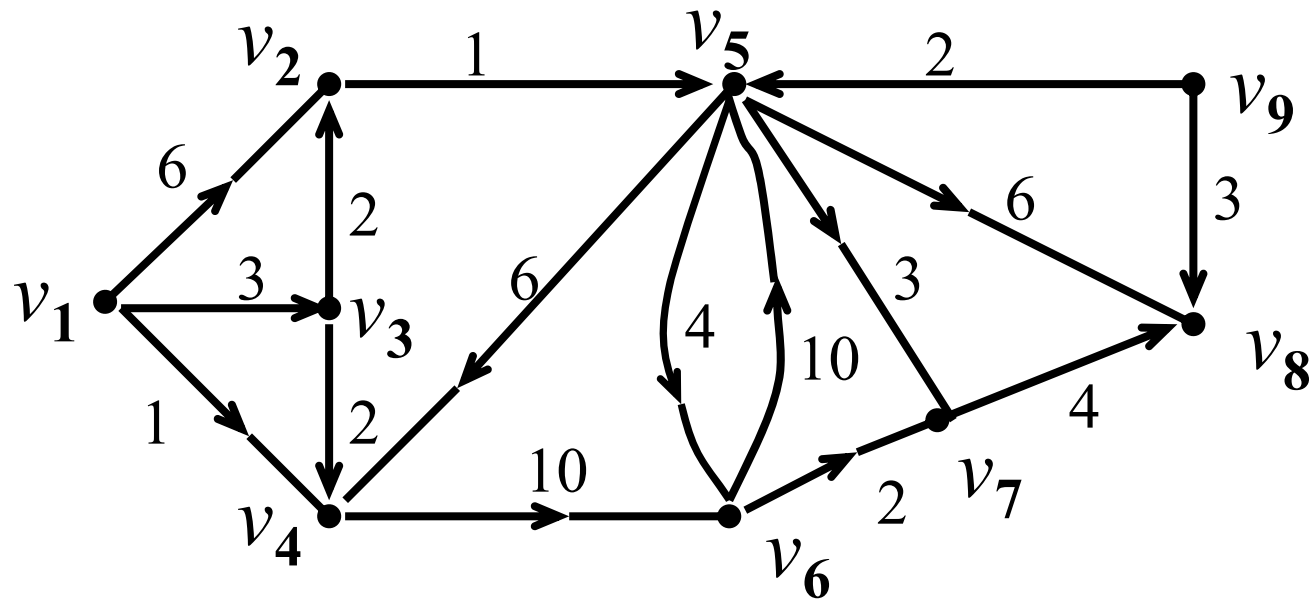
#### 输出参数:

- (1) P – 最短路径经过的节点
- (2) d – 最短距离

## 课后作业

### 论文作业（提交时请附上单独的代码文件打包一起提交）

下图为单行线交通网，每弧旁的数字表示通过这条线所需的费用。现在某人要从 $v_1$ 出发，通过这个交通网到 $v_8$ 去，求使总费用最小的旅行路线。



答案：对应的最短距离为12，路径为： $v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5 \rightarrow v_8$